PACKT PUBLISHING

掌握和使用正确的数据可视化方法

Python 数据可视化编程实战

Python Data Visualization Cookbook

[爱尔兰] Igor Milovanović 著

颛清山 译

人民邮电出版社
POSTS & TELECOM PRESS

□ □

Python[illegible]

[[illegible]]Igor Miloveanović [illegible]

[illegible] [illegible]

[illegible]

[illegible]

[illegible]**CIP**[illegible]

Python[illegible]/[illegible]Milovanovic,I.[illegible].--[illegible]2015.5

ISBN 978-7-115-38439-3

Ⅰ.①P… Ⅱ.①[illegible]…②[illegible]… Ⅲ.①[illegible]—[illegible] Ⅳ.①TP311.56

[illegible]CIP[illegible]2015[illegible]057566[illegible]

[illegible]

◆[illegible] [[illegible]]Igor Milovanović

[illegible] [illegible]

[illegible] [illegible]

[illegible] [illegible] [illegible]

◆[illegible] [illegible]11[illegible]

[illegible] 100164 [illegible] 315@ptpress.com.cn

[illegible] http://www.ptpress.com.cn

[illegible]

◆[illegible]800×1000 1/16

[illegible]16.75

[illegible]318[illegible] 2015[illegible]5[illegible]1[illegible]

[illegible]1-3000[illegible] 2015[illegible]5[illegible]1[illegible]

[illegible] [illegible]01-2013-9037[illegible]

[illegible]49.00[illegible]

[illegible]010[illegible]81055410　[illegible]010[illegible]81055316

[illegible]010[illegible]81055315

[illegible]

[illegible] Python [illegible] Python [illegible]60[illegible]

[illegible]8[illegible]3D [illegible]matplotlib[illegible]

[illegible]Python[illegible]Python[illegible]

[illegible]

[illegible]Python[illegible]matplotlib[illegible]2D[illegible]3D[illegible]matplotlib[illegible]matplotlib[illegible]matplotlib[illegible]

matplotlib [illegible] John Hunter [illegible] matplotlib [illegible]John Hunter [illegible]John Hunter [illegible]MATLAB[illegible]John Hunter[illegible]MATLAB[illegible]MATLAB[illegible]Python[illegible]matplotlib[illegible]

[illegible]matplotlib[illegible]MATLAB[illegible]MATLAB[illegible]MATLAB[illegible]matplotlib[illegible]

matplotlib [illegible] Python [illegible] shell [illegible]matplotlib[illegible]png[illegible]pdf[illegible]ps[illegible]eps[illegible]svg[illegible]

[illegible]Python[illegible]Locust[illegible] matplotlib [illegible]

[illegible]matplotlib[illegible]

[illegible]cookbook[illegible]“do the right thing in the right way”[illegible]

[illegible]MATLAB[illegible]1[illegible]

[illegible]zhuanqingshan@163.com[illegible]

[illegible]

2014[illegible]12[illegible]

[illegible]

Igor Milovanović [illegible]Linux[illegible]

[illegible]

[illegible]

[illegible]

[illegible]Python[illegible]matplotlib[illegible]

□□□□

Tarek Amr □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□ 10 □□□□□ 2007 □□□□□□ Global Voices Online (GVO)□□ □□□□□□□ Open Knowledge Foundation (OKFN)□□□□□□□□ □□Government 2.0□□□□□□□□□□□□□□□□□□□

Tarek □ Twitter □ □ □ @gr33ndata □ □ □ □ http://tarekamr.appspot.com/□

Jayesh K. Gupta □ Matlab Toolbox for BiclusteringAnalysis (MTBA)□□□□□□□□□□□IIT Kanpur□□□□ □□ IIT□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□ rejuvyesh□□□□□□□□□□□□Goodreads□□□□□□□□□□□□□□ Bitbucket□GitHub□□□□□□□□□□□□□http://home.iitk.ac.in/□ jayeshkg/□□□□□□a2z.jayesh@gmail.com□□□

Kostiantyn Kucher □□□□□□□□2012 □□□□□□□□□□□ □□□□□□□□□□□□□Python□Matplotlib□PIL□□□□□□□□□□□

□□□Kostiantyn□□□□□□□□□□□□□□□□□□□□□□Andreas Kerren□□□□□□□□□□□□□□□□ISOVIS□□□□□

Kenneth Emeka Odoh □□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□

Kenneth [illegible] Python [illegible]2012 [illegible] Pycon [illegible] Django[illegible]

[illegible] C[illegible]C++[illegible]Python[illegible]Java [illegible]

[illegible]Kenneth[illegible]

[illegible]

[illegible]

[illegible]CSV[illegible]JSON[illegible]

[illegible]matplotlib[illegible]Python[illegible] Python [illegible]

[illegible]Python[illegible]matplotlib[illegible]

[illegible] Ubuntu 12.03[illegible] Python 2.7[illegible]IPython 0.13.2[illegible]virtualenv 1.9.1[illegible]matplotlib 1.2.1[illegible]NumPy 1.7.1 [illegible] SciPy 0.11.0[illegible]

[illegible]

[illegible]1[illegible]Python[illegible]

[illegible]2[illegible]CSV[illegible]JSON[illegible]XSL[illegible]

[illegible]3[illegible]

[illegible]4[illegible]

[illegible]5[illegible]3D[illegible]3D[illegible]3D[illegible]matplotlib[illegible]

[illegible]6[illegible]CAPTCHA[illegible]

[illegible]7[illegible]

[illegible]8[illegible]matplotlib[illegible]matplotlib[illegible]LaTeX[illegible]

[illegible]

[illegible]Python2.7.3[illegible]Ubuntu12.03[illegible]Python[illegible]2.7.3[illegible]

[illegible]IPython[illegible]Python[illegible]Linux[illegible]Windows[illegible]Mac OS[illegible]

[illegible]Python[illegible]Python[illegible]Anaconda[illegible]Enthought Python[illegible]Python[illegible]X,Y[illegible]

[illegible]Python[illegible]Python[illegible]pip[illegible]pip[illegible]Python[illegible]easy_install[illegible]

[illegible]

[illegible]Python[illegible]Python[illegible]

[illegible]

[illegible]

[illegible]

[illegible]“[illegible]DemoPIL[illegible]run_fixed_filters_demo[illegible]”

[illegible]

```
def _load_image(self, imfile):
    self.im = mplimage.imread(imfile)
```

[illegible]

```
# tidy up tick labels size
all_axes = plt.gcf().axes
for ax in all_axes:
    for ticklabel in ax.get_xticklabels() + ax.get_yticklabels():
        ticklabel.set_fontsize(10)
```

[illegible]

```
$ sudo python setup.py install
```

[illegible] “[illegible] **0** [illegible]”

[illegible]

[illegible]

[illegible]

[illegible]——[illegible]

[illegible]feedback@packtpub.com[illegible]

[illegible]
www.packtpub.com/authors[illegible]

[illegible]

[illegible]Packt[illegible]

[illegible]

[illegible] http://www.packtpub.com [illegible]http://www.packtpub.com/support [illegible]

[illegible]

[illegible]——[illegible]——[illegible]
http://www.packtpub.com/submit-errata[illegible]
errata submission form [illegible]
http://www.packtpub.com/support[illegible]

[illegible]

[illegible]Packt[illegible]

[illegible]copyright@packtpub.com[illegible]

[illegible]

[illegible]

[illegible] questions@packtpub.com [illegible]

[illegible]1[illegible] [illegible]

[illegible]

◆ [illegible]matplotlib[illegible]NumPy[illegible]SciPy[illegible]

◆ [illegible]virtualenv[illegible]virtualenvwrapper

◆ [illegible]Mac OSX[illegible]matplotlib

◆ [illegible] Windows [illegible] matplotlib

◆ [illegible]Python[illegible]Python Imaging Library[illegible]PIL[illegible]

◆ [illegible]requests[illegible]

◆ [illegible]matplotlib[illegible]

◆ [illegible]matplotlib[illegible]

1.1 [illegible]

[illegible]Python[illegible]

1.2 [illegible]matplotlib[illegible]Numpy[illegible]Scipy[illegible]

[illegible]matplotlib[illegible]Linux[illegible]

1.2.1 □□□□

□□□□□□□□□□□□Linux □□□□□□□□Python □□□□□□Debian/Ubuntu □RedHat/SciLinux□□□□□□□□□□Linux□□□□□□□□Python□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Python□□□□□□□□□□□□□□□□□□□□Python□□□2.7□□□□

□□□□□□□□□□□□ Python 3.3 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Python□□□□□□2.7□□□□□2.6□□□□□□□□□Python 2.7 □□□□□□□□ Python □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Python3.3□□□□□□□□□□range□□□□□xrang□□□□□

□□

□□matplotlib□□□□□□□□□□□□□□□□□□□

Matplotlib□□□□□□□□□NumPy□libpng□freetype□□□□□□□□□□□□□□matplotlib□□□□□□□□□□NumPy□□□□□□□□□□http://www.numpy.org/□□□□□NumPy□□□□□□□□□□□1.4□□□□□□□□Python 3□□□NumPy 1.5□□□□□□□□

NumPy□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Python□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□NumPy□□□□□□□□“□□”□□□C□□□□□□□□□□□□□□□□□□□□

□□NumPy□□□□SciPy□□□Python□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Netlib □ □ □ □ □ □ □ http://www.netlib.org□□□□□□□□□□C□□□□Fortran□□□□□□□

□□NumPy□□□□□□□

1.□□Python-NumPy□□□□

```
$ sudo apt-get install python-numpy
```

2.□□□□□□□□

```
$ python -c 'import numpy; print numpy.__version__'
```

3.□□□□□□□

◆ libpng 1.2□PNG □□□□□□□ zlib □□□

◆ freetype 1.4+□□□ True type □□□

```
$ sudo apt-get install build-dep python-matplotlib
```

□□□□RedHat□□□RedHat□Linux□□□□Fedora□SciLinux□CentOS□□□□□□yum□□□□□□□□□□apt-get□□□□□

```
$ su -c 'yum-builddep python-matplotlib'
```

1.2.2 □□□□

□□matplotlib□□□matplotlib□python□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□Ubuntu□□□□□□□□□□□□□□□□□□□□□

```
# in your terminal, type:
$ sudo apt-get install python-numpy python-matplotlib python-scipy
```

□□□

□□□□□□□□□www.github.com□□□□□□□□□□□□□□□□□□

```
$ cd □/Downloads/
$ wget https://github.com/downloads/matplotlib/matplotlib/matplotlib-1.2. 0.tar.gz
$ tar xzf matplotlib-1.2.0.tar.gz
$ cd matplotlib-1.2.0
$ python setup.py build
$ sudo python setup.py install
```

4 Python □□□□□□□□□

□□□□□□

□□□□□□□□□ http://www.packtpub.com □□□□□□ Packt □□□http://www.packtpub.com/support/□□□□□□□□□□□□□□□□□□□□□□□

1.2.3 □□□□

□□□□□□ matplotlib, □□□□□□ Python □□□□ Distutils□□□□□□□□□□□□□□□□□□□□□□□□□ Linux □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

1.2.4 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□IPython□□□□□□□□IPython□□□□□Python□□□□□□□□□□□PyLab□□□□□□□□□matplotlib□□□□□□□□□□□□NumPy□

SciPy□□□□□□□□□□□□□□□□IPython □□□□□□□□□□□□□□□□□□□□□□□□
IPython □□□□□□□□□□□□

1.3 □□virtualenv□ virtualenvwrapper

□□virtualenv□

virtualenv □□ Ian Bicking □□□□□□□□□□□□□□□□□, □□

□□□□□Django □□□□□□□ Django 1.1 □ Python 2.3 □□□□□□□□□□□□□□□□□□□□□□□Python2.6□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Python□□□□□□□□□□□□□□□□□

virtualenv□□□, □ virtualenv □□□□□□□□□□□□□□□□□□□□

1.3.1 □□□□

□□□virtualenv□□□□□Python□pip□Pip□□□□□□Python□□□□□□□□□□□□□□□ easy install □□□□□□□□□□□□□□□□ pip □□□□□□□□□□□□□□□root□□□□□□□□□□□□□□□□□□pip□□□□

```
# easy_install pip
```

virtualenv □□□□□□□□□□□□□□□□ virtualenvwrapper□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□virtualenvwrapper □□□□□□ http://virtualenvwrapper.
readthedocs.org/en/latest/#features□

1.3.2 □□□□

□□virtualenv□virtualenvwrapper□□□□□□□□

1.□□virtualenv□virtualenvwrapper□

$ sudo pip virtualenv

$ sudo pip virtualenvwrapper

□□□□□□□□□□□□□□ export □□□□□□□

$ export VIRTENV=□/.virtualenvs

$ mkdir -p $VIRTENV

# □□ source □□□□□□□□shell □□□□□□□

$ source /usr/local/bin/virtualenvwrapper.sh

□□□□□□□□

$ mkvirtualenv virt1

2.□virt1□□□□□matplotlib□

(virt1)user1:□$ pip install matplotlib

3.□□□□□□□□□□□□□□□/.bashrc□□

source /usr/loca/bin/virtualenvwrapper.sh

□□□□□□□□□□□□□□□□

◆ mkvirtualenv ENV: □□□□ ENV □□□□□□□□□

◆ workon ENV: □□□□□□□ ENV □□□□□

◆ deactivate: □□□□□□□□□

1.4 □Mac OS X□□□matplotlib

□ Mac OS X □□□ matplotlib □□□□□□□□□□□□□ python □□□□□□□Enthought Python Distribution (EPD)□□□□□□□□□ EPD □□□□□□□□□□□□□□□□□□□□

□□EPD□□Python□matplotlib□□□□□□

1.4.1 □□□□

□□Apple□□□□□□□□□□□□□□□□□Homebrew□□□□□□□□□□□□□□□□□Homebrew□□□Ruby□Git□□□□□□□□□□□□□□□□□□□□□□□□□□□□Homebrew,□□□□ Python□□□□□□□□ virtualenv □□□□□□□□□□□□matplotlib□□□□NumPy□SciPy□□□□□□matplotlib□□□□□□□□□

1.4.2 □□□□

1.□□□□□□□□□□□□□□□

```
ruby <(curl -fsSkL raw.github.com/mxcl/homebrew/go)
```

□□□□□□□□□□□□□ brew update □ brew doctor □□□□□ brew □□□□□□□□□

2.□□□□Homebrew□□□□□□□path□□□□□□□□□□□Homebrew□□□□□□□□□□□□□□□□□□□□□□□□□/.bash_profile□□□□□/Users/[your-user-name]/.bash_profile□□□□□□□□□□□□□□

```
export PATH=/usr/local/bin:$PATH
```

3.□□□□□□□□□□□□□□□□ path □□□□□□□□□□□□□□□□□□□□□□□□Python□□□□

```
brew install python --framework –universal
```

□□□□□□□□□Python□□□□□□□□

4.□□path□□□□□□□□□□□□□

export
PATH=/usr/local/share/python:/usr/local/bin:$PATH

5.□□□□□□ python –version,□□ python □□□□□□□

□□□□□□□□□□Python□□□□□2.7.3□

6.pip□□□□□□□□□□□□□□□□□□□easy_install□□pip□

$ easy_install pip

7.□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ virtualenv □ virtualenvwrapper□

pip install virtualenv

pip install virtualenvwrapper

8.□□□□□□□□□□□□□□——□□matplotlib□

pip install numpy

brew install gfortran

pip install scipy

Mountain Lion □□□□□□□ SciPy □□□□□0.11□□□□□□□

pip install -e git+https://github.com/scipy/scipy#egg=scipy- dev

9.□□□□□□□□□□□Python□□□□□□□□

import numpy

print numpy.__version__

import scipy

print scipy.__version__

```
quit()
```

10.□□matplotlib□

```
pip install matplotlib
```

1.5 □Windows□□□matplotlib

□□□□□□□□□□□□□□□Python□matplotlib□□□□□□□□□□□□□Python□

1.5.1 □□□□

□Windows□□□matplotlib□□□□□□□□□□□□□□□□□□Python□□□□EPD□Anaconda□Python(x,y)□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□matplotlib□□□□□□□□□□□□□□NumPy□SciPy□□□□□□□□□□□□□□□□matplotlib□□□matplotlib□□□□□□□□□□□□□□□

1.5.2 □□□□

□□□□□□□Python□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□matplotlib,□□□□Python□□□□□□□□□□□□□□□□□□□Enthought Python Distribution(EPD)□□□□□ matplotlib □□□□□□□□□□□□□□□□□SciPy□NumPy□IPython□□□□□□□□□□□□□□□□EPD□□□□

matplotlib □□□□□□□□□□□□□□□□□□□□□ Windows Installer □□□□□*.exe□□□□□□□□

Python(x,y) □ http://code.google.com/p/pythonxy/ □ □ □ □ Windows 32 □□□□□□□□□□□□□□□□□□ matplotlib □□□□□□□□□□□□□ Windows □□□□□□matplotlib □□□□□□□□□□□□□□□□□□□□□□□□□Python(x,y)□ Python □□□□□□□□□□□□□□□□□□Python(x,y)□□□□□□□□□Python□□□□□Python(x,y)□□□□□□□□□□□Python□

□□□□□□□□□□□□□□□□□□Python□NumPy□SciPy□matplotlib□□□□□□□matplotlib□□□□□□□□□□□□MSI□□□□□□□□□□□x86□x86-64□□□□Python□□□□□□□□□NumPy□SciPy□□□□□□□□□□□□□□□□□□□□□NumPy□SciPy□□□□□□□□□□□□□matplotlib□□□□□□□□□□□□□□□□□□□□

1.5.3 □□□□

□□□□□Windows□□□□□matplotlib□□□□□□□□□□□□□□□□□□□□□□□□□□□matplotlib□□□□□□examples□□□□

1.6 □□□□□□□□□□Python□□□□PIL□

Python□□□□PIL□□Python□□□□□□□□□□PIL□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□point operations□□□□□filtering□□□□□□□□□□□□□□□□□arbitrary affine transforms□□ PIL □□□□□□□□□□□□□□□□□□□□□□□□□□□histogram□□□□□

PIL □□

PIL [illegible]

1.6.1 [illegible]

[illegible]

[illegible]Debian/Ubuntu[illegible]

```
$ sudo apt-get build-dep python-imaging
$ sudo pip install http://effbot.org/downloads/Imaging-1.1.7.tar.gz
```

1.6.2 [illegible]

[illegible]apt-get[illegible]PIL[illegible]pip[illegible]PIL[illegible]Ubuntu[illegible]PIL[illegible]

[illegible]RedHat/SciLinux[illegible]

```
# yum install python-imaging
# yum install freetype-devel
# pip install PIL
```

1.6.3 [illegible]

[illegible] PIL [illegible] http://www.pythonware. com/library/pil/handbook/index.htm [illegible] PDF [illegible] http://www.pythonware. com/media/data/pil-handbook.pdf[illegible]

Pillow [illegible] PIL [illegible]Pillow [illegible]http://pypi.python.org/pypi/Pillow[illegible]

[illegible] Windows [illegible] PIL [illegible] http://www.pythonware. com/products/pil/[illegible].exe[illegible] PIL [illegible] Python [illegible] site-packages[illegible]

[illegible] PIL [illegible] PIL.pth [illegible] C:\Python27\Lib\site-packages [illegible] PIL [illegible] virtualenv [illegible] site-packages[illegible]

1.7 [illegible]requests[illegible]

[illegible]HTTP[illegible] Python[illegible]requests[illegible]

[illegible]Python[illegible]urllib2[illegible]HTTP[illegible]

Request[illegible]API[illegible]Web[illegible]Requests [illegible] HTTP 1.1 [illegible]

1.7.1 [illegible]

[illegible]requests[illegible]pip[illegible]

```
$ pip install requests
```

[illegible]virtualenv[illegible]requests[illegible]requests[illegible]

[illegible]requests[illegible]requests[illegible]

```
import requests
r = requests.get('http://github.com/timeline.json')
print r.content
```

1.7.2 requests [illegible]

[illegible] www.github.com [illegible] URI [illegible] HTTP GET [illegible] JSON [illegible] GitHub [illegible] https://github.com/timeline [illegible]HTML[illegible]HTTP[illegible]r[illegible]HTTP[illegible]HTTP[illegible]cookies[illegible]HTTP[illegible]

1.8 [illegible]matplotlib[illegible]

matplotlib[illegible]Python[illegible].rc[illegible]matplotlib[illegible]

1.8.1 [illegible]

[illegible]matplotlib[illegible]matplotlib[illegible]

1.8.2 [illegible]

[illegible]rcParams[illegible] matplotlib.rc()[illegible]rcParams[illegible] matlotlib.rc()[illegible]

[illegible]matplotlib.rcdefaults()[illegible]

[illegible]

[illegible]matplotlib.rcParams[illegible]

```
import matplotlib as mp
mpl.rcParams['lines.linewidth'] = 2
mpl.rcParams['lines.color'] = 'r'
```

[illegible]matplotlib.rc()[illegible]

```
import matplotlib as mpl
mpl.rc('lines', linewidth=2, color='r')
```

[illegible]2[illegible]

```
import matplotlib.pyplot as plt
import numpy as np
t = np.arange(0.0, 1.0, 0.01)
s = np.sin(2 * np.pi * t)
# make line red
plt.rcParams['lines.color'] = 'r'
plt.plot(t,s)
c = np.cos(2 * np.pi * t)
# make line thick
plt.rcParams['lines.linewidth'] = '3'
plt.plot(t,c)
plt.show()
```

1.8.3 [illegible]

[illegible]matplotlib.pyplot[illegible]NumPy[illegible] plt.rcParams['lines.color']= 'r'[illegible]plt.rcParams['lines. linewidth'] = '3'[illegible] 3 [illegible]

[illegible]matplotlib.rcdefaults()[illegible]

1.9 [illegible]matplotlib[illegible]

[illegible]matplotlib[illegible]

1.9.1 [illegible]

[illegible] matplotlib [illegible]

1.9.2 [illegible]

[illegible]matplotlib[illegible]matplotlib[illegible]

[illegible] matplotlibrc [illegible]matplotlib [illegible]
matplotlibrc[illegible]

1.9.3 [illegible]

[illegible]

◆ [illegible]
matplotlib[illegible]matplotlibrc[illegible]

◆ [illegible] .matplotlib/matplotlibrc [illegible] (Per user .matplotlib/matplotlibrc)[illegible]$HOME [illegible] Windows [illegible]

[illegible] Documents and Settings [illegible] matplotlib.get_configdir()[illegible]

◆ [illegible]Per installation configuration file[illegible] python[illegible]site-packages[illegible]matplotlib[illegible]

[illegible]shell[illegible]

$ python -c 'import matplotlib as mpl; print mpl.get_configdir()'

[illegible]

◆ axes[illegible]

◆ backend[illegible] TkAgg[illegible] GTKAgg[illegible]

◆ figure[illegible] dpi[illegible]subplot[illegible]

◆ font[illegible]font family[illegible]

◆ grid[illegible]

◆ legend[illegible]

◆ line[illegible]

◆ patch[illegible] 2D [illegible]

◆ savefig[illegible]

◆ text[illegible] latex [illegible]

◆ verbose[illegible] matplotlib [illegible] silent[illegible]helpful[illegible] debug [illegible]debug-annoying[illegible]

◆ xticks [illegible] yticks[illegible] x[illegible]y [illegible]

1.9.4 [illegible]

[illegible] matplotlib [illegible] API [illegible]

[illegible]2[illegible] [illegible]

[illegible]

- ◆ [illegible] CSV [illegible]
- ◆ [illegible] Microsoft Excel [illegible]
- ◆ [illegible]
- ◆ [illegible]
- ◆ [illegible] JSON [illegible]
- ◆ [illegible] JSON[illegible]CSV [illegible] Excel
- ◆ [illegible]
- ◆ [illegible]
- ◆ [illegible]
- ◆ [illegible]
- ◆ [illegible] NumPy [illegible]
- ◆ [illegible]
- ◆ [illegible]

2.1 [illegible]

[illegible]

2.2 [illegible]CSV[illegible]

[illegible]——CSV[illegible]CSV[illegible]

Python[illegible]csv[illegible]CSV[illegible]CSV[illegible]CSV[illegible]

2.2.1 [illegible]

[illegible]ch02-data.csv[illegible]

2.2.2 [illegible]

[illegible]CSV[illegible]

1.[illegible]ch02-data.csv[illegible]

2.[illegible]

3.[illegible]

4.[illegible]

[illegible]

```
import csv
filename = 'ch02-data.csv'
data = []
try:
    with open(filename) as f:
        reader = csv.reader(f)
    header = reader.next()
```

```
        data = [row for row in reader]
    except csv.Error as e:
        print "Error reading CSV file at line %s: %s" %
(reader.line_num, e)
        sys.exit(-1)
    if header:
        print header
        print '=================='
    for datarow in data:
        print datarow
```

2.2.3 □□□□

□□□□□csv□□□□□□□□□□□□□□□□□with□□□□□□□□□□□□□□□□f□□□□□□□□□□□□□□□□□□with□□□

□□□□csv.reader()□□□□reader□□□

□□CSV□□□CSV□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Linux□□□□bash□□□head□□□□□□□□□□□□□□□

$ head some_file.csv

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□csv.reader()□□□

2.2.4 □□□□

□□□□□csv□□□□□□□□□□□□□□PEP□□□□□CSV□□API□□□□ http://www.python.org/dev/peps/pep-0305/□

□□□□□□□□□□□□□□□□□□□□□□□□NumPy□loadtxt()□□□□□□□□□□□□CSV□□□□□

□□□□□□□□□□□□□□□□□

```
import numpy
data=numpy.loadtxt('ch02-data.csv',dtype='string',
delimiter=',')
```

□□□□□□□□□□NumPy□□□□□□□□□□□□□□□numpy.loadtxt()□□□□□numpy.genfromtxt()□□□

□□□□ Python 2.7.x □□□□csv□□□□ Unicode □□□□□□□□□□□□□□□□□□□□UTF-8□□ASCII□□□□□Python CSV□□□□□□□□□□□□□□□□□□□

Python3.3□□□□□□□□Unicode□□□□□□□□□□

2.3 □Microsoft Excel□□□□□□

□□ Microsoft Excel □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Python□□□□□□□

□Excel□□□□□□□□□□□□□□Excel□□□□ CSV□□□□□□□□□□□□□□□□□□Python□CSV□□□□□□□□□□□□□□□□□□□□□□□ Microsoft

Excel [illegible] OpenOffice.org[illegible]Excel[illegible]CSV[illegible]Excel[illegible]

[illegible]www.python-excel.org[illegible]Python[illegible]Excel[illegible]Excel[illegible]Windows[illegible]

Microsoft Excel [illegible] Python [illegible]XLRD[illegible]0.90[illegible].xlsx[illegible]

2.3.1 [illegible]

[illegible]xlrd[illegible]pip[illegible]

```
$ mkvirtualenv xlrdexample
(xlrdexample)$ pip install xlrd
```

[illegible]ch02-xlsxdata.xlsx[illegible]

2.3.2 [illegible]

[illegible]Excel[illegible]

1.[illegible]

2.[illegible]nrows[illegible]ncols[illegible]

3.[illegible]

```
import xlrd
file = 'ch02-xlsxdata.xlsx'
wb = xlrd.open_workbook(filename=file)
ws = wb.sheet_by_name('Sheet1')
dataset = []
```

```
for r in xrange(ws.nrows):
    col = []
    for c in range(ws.ncols):
        col.append(ws.cell(r, c).value)
    dataset.append(col)
from pprint import pprint
pprint(dataset)
```

2.3.3 [illegible]

[illegible] xlrd [illegible]xlrd.sheet.Sheet[illegible]Python [illegible] xlrd.book.Book[illegible]xlrd.sheet.Cell[illegible]

[illegible]open_workbook()[illegible] xlrd.book [illegible]Book [illegible] sheet_by_name()[illegible]sheets()[illegible]sheets()[illegible] xlrd.sheet.Sheet [illegible]xlrd.sheet.Sheet[illegible]cell()[illegible]xrld.sheet.Cell[illegible]

[illegible]xlrd [illegible] Python date [illegible]xlrd [illegible] xlrd.XL_CELL_DATE[illegible]

```
from datetime import datetime
from xlrd import open_workbook, xldate_as_tuple
...
cell = sheet.cell(1, 0)
```

```
print cell
print cell.value
print cell.ctype
if cell.ctype == xlrd.XL_CELL_DATE:
    date_value = xldate_as_tuple(cell.value, book.datemode)
    print datetime(*date_value)
```

[illegible]

2.3.4 [illegible]

xlrd [illegible] open_workbook[illegible]on_demand[illegible]True[illegible]

```
book = open_workbook('large.xls', on_demand=True)
```

[illegible] Excel [illegible] Excel[illegible]——xlwt[illegible]“[illegible]JSON[illegible]CSV[illegible]Excel”[illegible]

[illegible]PyPi [illegible] Python [illegible] http://pypi.python.org/pypi?:action=browse&c=377[illegible]

2.4 [illegible]

[illegible] CSV[illegible]

[illegible]

[illegible]Python[illegible]struct[illegible]http://docs.python.org/library/struct.html[illegible]C[illegible]Python[illegible]

2.4.1 [illegible]

[illegible]struct[illegible]Python[illegible]

2.4.2 [illegible]

[illegible]

...

207152670 3984356804116 9532

427053180 1466959270421 5338

316700885 9726131532544 4920

138359697 3286515244210 7400

476953136 0921567802830 4214

213420370 6459362591178 0546

...

[illegible]ch02-generate_f_data.py[illegible]

[illegible]

1.[illegible]

2.[illegible]

3.[illegible]

4.[illegible]

```
import struct
import string
datafile = 'ch02-fixed-width-1M.data'
# this is where we define how to
# understand line of data from the file
mask='9s14s5s'
with open(datafile, 'r') as f:
   for line in f:
      fields = struct.Struct(mask).unpack_from(line)
      print 'fields: ', [field.strip() for field in fields]
```

2.4.3 □□□□

□□□ head□more □□□□□ Linux shell □□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□

□□
9s15s5s□□□□□□□“9 □□□□□□□□□□□□□□ 15□□□□□□□□□□□□□□□5□□
□□□□□□□□”

□□□□□c□□□□□□C□□□□char□□□□□□□□1□□□□□s□□□□□□□C□□□□
char[]□□□□d□□□□□□□C□□□□double□□□□□□□□□□ Python □□□□□□
□ □ □ □ □ □ □ □ □ http://docs.python.org/library/struct.
html#format-characters□

□□□□□□□□□□□□□□□□□□□□□□□unpack_from□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□strip()□□□□□□□□□□□□□□□□□□□

□□□□□□□□□ struct.Struct □□□□□□□object-oriented, OO□□□
□□□□□□□□□□□□□□□□□□□□□□□□□

```
fields = struct.unpack_from(mask, line)
```

[illegible]struct.Struct[illegible]

2.5 [illegible]

[illegible]flat datafile[illegible]Excel[illegible]

[illegible]CSV[illegible]Python[illegible]csv[illegible]——[illegible]

2.5.1 [illegible]

[illegible]CSV[illegible]2.2“[illegible]CSV[illegible]”[illegible]

2.5.2 [illegible]

[illegible]2.2“[illegible]CSV[illegible]”[illegible]

```
import csv
filename = 'ch02-data.tab'
data = []
try:
    with open(filename) as f:
        reader = csv.reader(f, dialect=csv.excel_tab)
    header = reader.next()
        data = [row for row in reader]
except csv.Error as e:
    print "Error reading CSV file at line %s: %s" % (reader.line_num, e)
    sys.exit(-1)
if header:
    print header
    print '====================='
for datarow in data:
    print datarow
```

2.5.3 □□□□

□□□□□ csv □□□□□□□□□□□□□□□□□“□ CSV □□□□□□”□□□□□□□□□□□□□□□□□□□dialect□□□excel_tab□□□

2.5.4 □□□□

□□CSV□□□□□□□□□□□□□□□□□“□□□”□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□\t□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ch02-data-dirty.tab□□□“□□□”□□□□□□□□□□□□□□□□□□□□□□□□□□□“□□□”□□□□□□

```
    datafile = 'ch02-data-dirty.tab'
    with open(datafile, 'r') as f:
        for line in f:
            # remove next comment to see line before cleanup
            # print 'DIRTY: ', line.split('\t')
            # we remove any space in line start or end
            line = line.strip()
            # now we split the line by tab delimiter
```

```
print line.split('\t')
```

[illegible]——[illegible] split('\t')[illegible]

[illegible]csv[illegible]split()[illegible] .csv[illegible].tab[illegible]csv.Sniffer[illegible]

2.6 [illegible]JSON[illegible]

[illegible] JSON [illegible]

JavaScript Object Notation[illegible]JSON[illegible]

[illegible]URL[illegible]/[illegible]JSON[illegible]

2.6.1 [illegible]

[illegible] requests [illegible]PYTHONPATH[illegible]1[illegible]“[illegible]”[illegible]

[illegible]

2.6.2 [illegible]

[illegible]GitHub[illegible]http://github.com[illegible]

1.[illegible] GitHub URL [illegible] JSON [illegible]

2.□□requests□□□□□□□URL□□□□□□□

3.□□□□□□□□□□JSON□□□□□□

4.□□□□JSON□□□□□□□□□□□□□□□□□□□□URL□□

```
import requests
url = 'https://github.com/timeline.json'
r = requests.get(url)
json_obj = r.json()
repos = set()
for entry in json_obj:
   try:
      repos.add(entry['repository']['url'])
   except KeyError as e:
      print "No key %s. Skipping..." % (e)
from pprint import pprint
pprint(repos)
```

2.6.3 □□□□

□□□□ requests □□□□□□□□□requests □□□□□□□□ API □□□HTTP□□□□□□□□□□□get()□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Response □□□□□□□□□□□□□□□□□□□□Response.json()□□□□□□□□□□□□□□Response.content□□□□□□□□□JSON□□□□JSON□□□□

□□□□JSON□□wget□curl□□JSON□□□□□□□□□

□□□□□□□IPython□□□□□□□□□□□□□□□□□□□□IPython□□□□%run program_name.py □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□%who□□%whos□□□□□□□

[illegible]JSON[illegible]

JSON[illegible]Python[illegible] key [illegible] entry['repository'] ['url'][illegible]URL[illegible]

[illegible]entry['repository'['ur'][illegible]JSON[illegible]

```
...
   "repository" : {
      ...
      "url" : "https://github.com/ipython/ipython",
      ...
   },
...
```

[illegible]Python[illegible]key[illegible]

2.6.4 [illegible]

JSON [illegible] RFC 4627 [illegible] http://tools.ietf.org/html/rfc4627. html[illegible] XML [illegible] JSON [illegible] JavaScript——[illegible]Web[illegible]

Python [illegible] JSON [illegible]JSONEncoder/JSONDecoder[illegible]Python[illegible]JSON[illegible]Python[illegible]JSON[illegible]

[illegible]JSONDecoder/JSONEncoder[illegible]

[illegible]json.loads()[illegible]Python[illegible]float[illegible]JSON[illegible]json[illegible]

[illegible]JSON[illegible]

```
jstring = '{"name":"prod1","price":12.50}'
```

[illegible]

```
from decimal import Decimal
json.loads(jstring, parse_float=Decimal)
```

[illegible]

```
{u'name': u'prod1', u'price': Decimal('12.50')}
```

2.7 [illegible]JSON[illegible]CSV[illegible]Excel

[illegible]

[illegible]Python[illegible]JSON[illegible]CSV[illegible]XLSX[illegible]

[illegible]“[illegible]”[illegible]

2.7.1 [illegible]

[illegible]Excel[illegible]xlwt[illegible]

```
$ pip install xlwt
```

2.7.2 [illegible]

[illegible]CSV[illegible]JSON [illegible] XLSX[illegible]

1.[illegible]

```
import os
import sys
import argparse
try:
    import cStringIO as StringIO
except:
    import StringIO
import struct
import json
import csv
2.□□□□□□□□□□□□□□□□
def import_data(import_file):
    '''
    Imports data from import_file.
    Expects to find fixed width row
    Sample row: 161322597 0386544351896 0042
    '''
    mask = '9s14s5s'
    data = []
    with open(import_file, 'r') as f:
        for line in f:
            # unpack line to tuple
            fields = struct.Struct(mask).unpack_from(line)
            # strip any whitespace for each field
            # pack everything in a list and add to full dataset
            data.append(list([f.strip() for f in fields]))
    return data
def write_data(data, export_format):
    '''Dispatches call to a specific transformer and returns data set.
    Exception is xlsx where we have to save data in a file.
    '''
    if export_format == 'csv':
        return write_csv(data)
    elif export_format == 'json':
        return write_json(data)
    elif export_format == 'xlsx':
        return write_xlsx(data)
    else:
        raise Exception("Illegal format defined")
```

3.□□□□□□□□□CSV□JSON□XLSX□□□□□□□□□□□□□

```
def write_csv(data):
    '''Transforms data into csv. Returns csv as string.
    '''
    # Using this to simulate file IO,
    # as csv can only write to files.
    f = StringIO.StringIO()
    writer = csv.writer(f)
    for row in data:
        writer.writerow(row)
    # Get the content of the file-like object
    return f.getvalue()
def write_json(data):
    '''Transforms data into json.Very straightforward.
    '''
    j = json.dumps(data)
    return j
def write_xlsx(data):
    '''Writes data into xlsx file.
    '''
    from xlwt import Workbook
    book = Workbook()
    sheet1 = book.add_sheet("Sheet 1")
    row = 0
    for line in data:
        col = 0
        for datum in line:
            print datum
            sheet1.write(row, col, datum)
            col += 1
        row += 1
        # We have hard limit here of 65535 rows
        # that we are able to save in spreadsheet.
        if row > 65535:
            print >> sys.stderr, "Hit limit of # of rows in one
    sheet (65535)."
            break
    # XLS is special case where we have to
    # save the file and just return 0
    f = StringIO.StringIO()
    book.save(f)
    return f.getvalue()
```

4.[illegible]main[illegible]

```
if __name__ == '__main__':
    # parse input arguments
    parser = argparse.ArgumentParser()
    parser.add_argument("import_file", help="Path to a
fixed-width data file.")
    parser.add_argument("export_format", help="Export
format: json, csv, xlsx.")
    args = parser.parse_args()
    if args.import_file is None:
        print >> sys.stderr, "You myst specify path to import
  from."
        sys.exit(1)
    if args.export_format not in ('csv','json','xlsx'):
        print >> sys.stderr, "You must provide valid export
  file format."
        sys.exit(1)
    # verify given path is accesible file
    if not os.path.isfile(args.import_file):
        print >> sys.stderr, "Given path is not a file:%s"%
  args.import_file
        sys.exit(1)
    # read from formated fixed-width file
    data = import_data(args.import_file)
    # export data to specified format
    # to make this Unix-lixe pipe-able
    # we just print to stdout
    print write_data(data, args.export_format)
```

2.7.3 [illegible]

[illegible]2.4“[illegible]”[illegible]stdout[illegible]

[illegible]JSON[illegible]CSV[illegible]XLSX[illegible]

[illegible] import_data()[illegible]Python[illegible]

[illegible]write_data()[illegible]write_csv()[illegible]

[illegible] CSV [illegible] csv.writer()[illegible]

[illegible]cat[illegible]

json[illegible]dump()[illegible]Python[illegible]JSON[illegible]CSV[illegible]stdout[illegible]

Excel[illegible]Excel[illegible]

[illegible] Book [illegible] stdout[illegible] Web service [illegible]

2.7.4 [illegible]

[illegible]

[illegible]write_data()[illegible]elif[illegible]write_*[illegible]

□□□□□□□□□□□□□□Python□□□

2.8 □□□□□□□

□□

□□□□ Python □□□□ SQL drivers □□□□

□□□□□SQLite□□□□□□□□□□□□□□□□□□□□□□SQL□□□□□□MySQL□PostgreSQL□□□□□□□□□□□□□□□SQL□□□□□□□□□□□□□SQL□□□□□□□□SQL□□□□□□□□□□□

2.8.1 □□□□

□□□□□□□□□□□□□□SQLite□□

```
$ sudo apt-get install sqlite3
```

Python□□□SQLite□□□□□□□□□□Python□□□□□□□□IPython□□□□□□□□□□□□□□□

```
import sqlite3
sqlite3.version
sqlite3.sqlite_version
```

□□□□□□□□□

```
In [1]: import sqlite3
In [2]: sqlite3.version
Out[2]: '2.6.0'
In [3]: sqlite3.sqlite_version
```

Out[3]: '3.6.22'

[illegible]sqlite3.version[illegible]Python[illegible]sqlite3[illegible]sqlite_version [illegible]SQLite[illegible]

2.8.2 [illegible]

[illegible]

1.[illegible]SQLite[illegible]

2.[illegible]

3.[illegible]

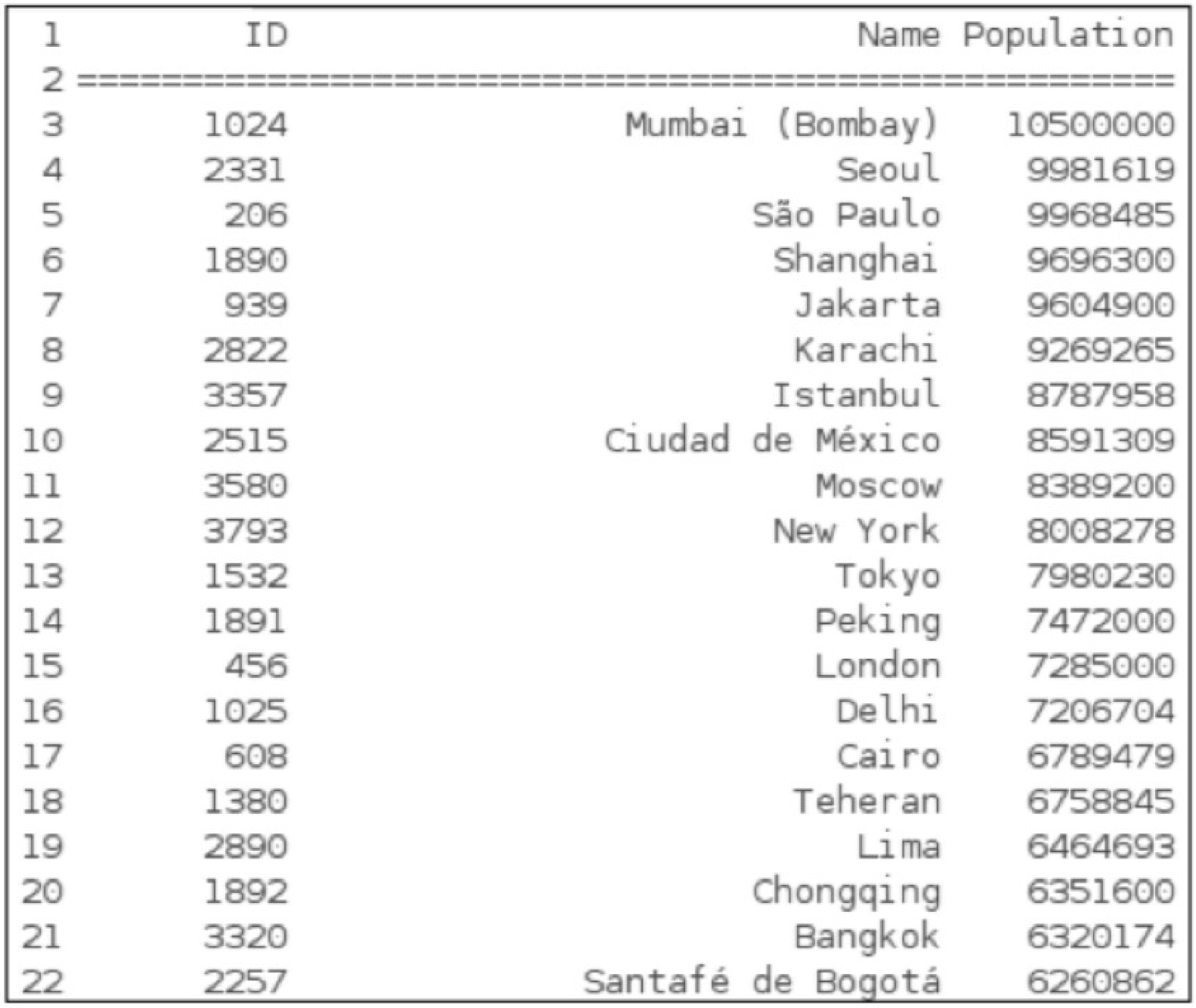
[illegible]SQL[illegible]SQL[illegible]

SELECT ID, Name, Population FROM City ORDER BY Population DESC LIMIT 1000

[illegible]City[illegible]ID[illegible]Name[illegible]Population[illegible]ORDER BY[illegible]Population[illegible]DESC[illegible]LIMIT[illegible]1000[illegible]

[illegible] world.sql [illegible]5000[illegible]

[illegible]2-1[illegible]



```
        ID                                   Name Population
=============================================================
      1024                        Mumbai (Bombay)   10500000
      2331                                  Seoul    9981619
       206                              São Paulo    9968485
      1890                               Shanghai    9696300
       939                                Jakarta    9604900
      2822                                Karachi    9269265
      3357                               Istanbul    8787958
      2515                       Ciudad de México    8591309
      3580                                 Moscow    8389200
      3793                               New York    8008278
      1532                                  Tokyo    7980230
      1891                                 Peking    7472000
       456                                 London    7285000
      1025                                  Delhi    7206704
       608                                  Cairo    6789479
      1380                                Teheran    6758845
      2890                                   Lima    6464693
      1892                              Chongqing    6351600
      3320                                Bangkok    6320174
      2257                    Santafé de Bogotá    6260862
```

[illegible]2-1

[illegible]SQL[illegible]SQLite[illegible]

```
import sqlite3
import sys
if len(sys.argv) < 2:
    print "Error: You must supply at least SQL script."
    print "Usage: %s table.db ./sql-dump.sql" % (sys.argv[0])
    sys.exit(1)
script_path = sys.argv[1]
```

```
if len(sys.argv) == 3:
    db = sys.argv[2]
else:
    # if DB is not defined
    # create memory database
    db = ":memory:"
try:
    con = sqlite3.connect(db)
    with con:
        cur = con.cursor()
        with open(script_path,'rb') as f:
            cur.executescript(f.read())
except sqlite3.Error as err:
    print "Error occured: %s" % err
```

□□□□□□□ SQL □□□□ SQL □□□□□□□□□ SQLite db □□□□□□□□□□□db□□□□SQLite□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□

```
import sqlite3
import sys
if len(sys.argv) != 2:
    print "Please specify database file."
    sys.exit(1)
db = sys.argv[1]
try:
    con = sqlite3.connect(db)
    with con:
```

```
    cur = con.cursor()
    query = 'SELECT ID, Name, Population FROM City ORDER BY Population DESC LIMIT 1000'
    con.text_factory = str
    cur.execute(query)
    resultset = cur.fetchall()
    # extract column names
    col_names = [cn[0] for cn in cur.description]
    print "%10s %30s %10s" % tuple(col_names)
    print "="*(10+1+30+1+10)
    for row in resultset:
        print "%10s %30s %10s" % row
except sqlite3.Error as err:
    print "[ERROR]:", err
```

2.8.3 [illegible]

[illegible]

[illegible]sqlite3.Error[illegible]

[illegible]con.cursor()[illegible]

[illegible]cur.fetchall()[illegible]fetchone()[illegible]

[illegible]cur.description[illegible]description[illegible]7[illegible]

[illegible]

2.8.4 [illegible]

[illegible]

[illegible] Python [illegible] MySQL[illegible]PostgreSQL[illegible]SQLite[illegible]MS SQL[illegible] Oracle [illegible] Sybase[illegible]Python [illegible]Oracle[illegible]PL/SQL[illegible]SQL[illegible] Oracle[illegible] MS SQL[illegible]SQLite [illegible] MySQL[illegible]MyISAM [illegible] InnoDB [illegible] SQL [illegible]http://en.wikipedia.org/wiki/SQL:2011[illegible]

2.9 [illegible]

[illegible]

[illegible]

[illegible][1][illegible]outlier[illegible]

2.9.1 [illegible]

[illegible]Python[illegible]

[illegible]——MAD[illegible]Median absolute deviation[illegible]MAD[illegible]

2.9.2 □□□□

□□□□□□□□MAD□□□□□□□□□□□□□□□□□□□

1.□□ 0□1 □□□□□□□□normally distributed random data□□

2.□□□□□□□□

3.□is_outlier()□□□□□□□□

4.□□□□□□□□□□x□filtered□□□□□□□□□□□□□

```
import numpy as np
import matplotlib.pyplot as plt
def is_outlier(points, threshold=3.5):
    """
    Returns a boolean array with True if points are outliers
and False otherwise.
    Data points with a modified z-score greater than this
    # value will be classified as outliers.
    """
    # transform into vector
    if len(points.shape) == 1:
        points = points[:,None]
    # compute median value
    median = np.median(points, axis=0)
    # compute diff sums along the axis
    diff = np.sum((points - median)**2, axis=-1)
    diff = np.sqrt(diff)
    # compute MAD
    med_abs_deviation = np.median(diff)
    # compute modified Z-score
```

```
        #
http://www.itl.nist.gov/div898/handbook/eda/section4/eda4
3.htm#
        # Iglewicz
        modified_z_score = 0.6745 * diff / med_abs_deviation
        # return a mask for each outlier
        return modified_z_score > threshold
    # Random data
    x = np.random.random(100)
    # histogram buckets
    buckets = 50
    # Add in a few outliers
    x = np.r_[x, -49, 95, 100, -100]
    # Keep valid data points
    # Note here that
    # "□" is logical NOT on boolean numpy arrays
    filtered = x[□is_outlier(x)]
    # plot histograms
    plt.figure()
    plt.subplot(211)
    plt.hist(x, buckets)
    plt.xlabel('Raw')
    plt.subplot(212)
    plt.hist(filtered, buckets)
    plt.xlabel('Cleaned')
    plt.show()
```

□□□□NumPy□□“□”□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□pylab□□□□□IPython□

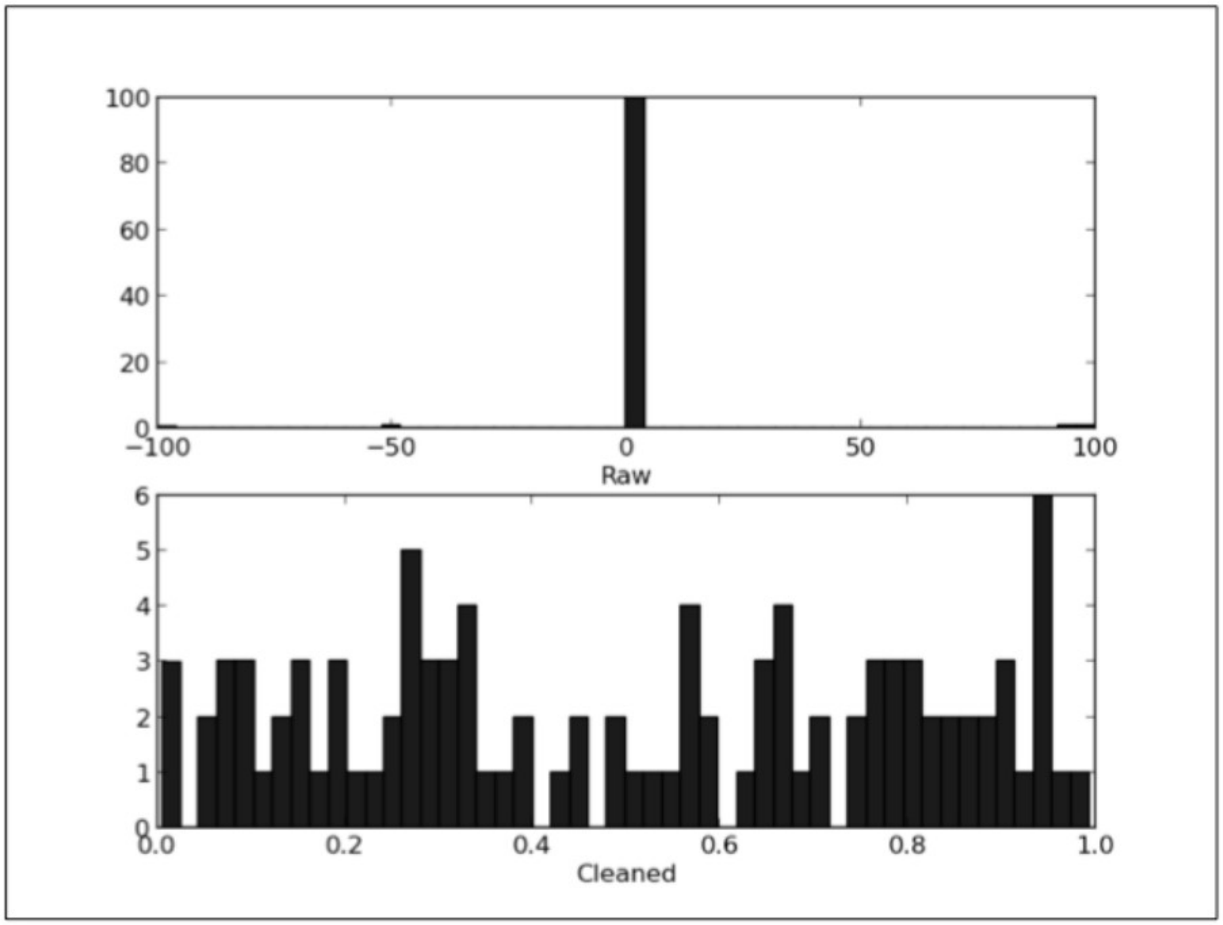
$ ipython –pylab

□□□□□□□

In [1]: □numpy.array(False)

Out[1]: True

□□2-2□□□
□□□□□□□□□□□□□□□□



□2-2

□□
□□□□□□□□□□box plot□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□

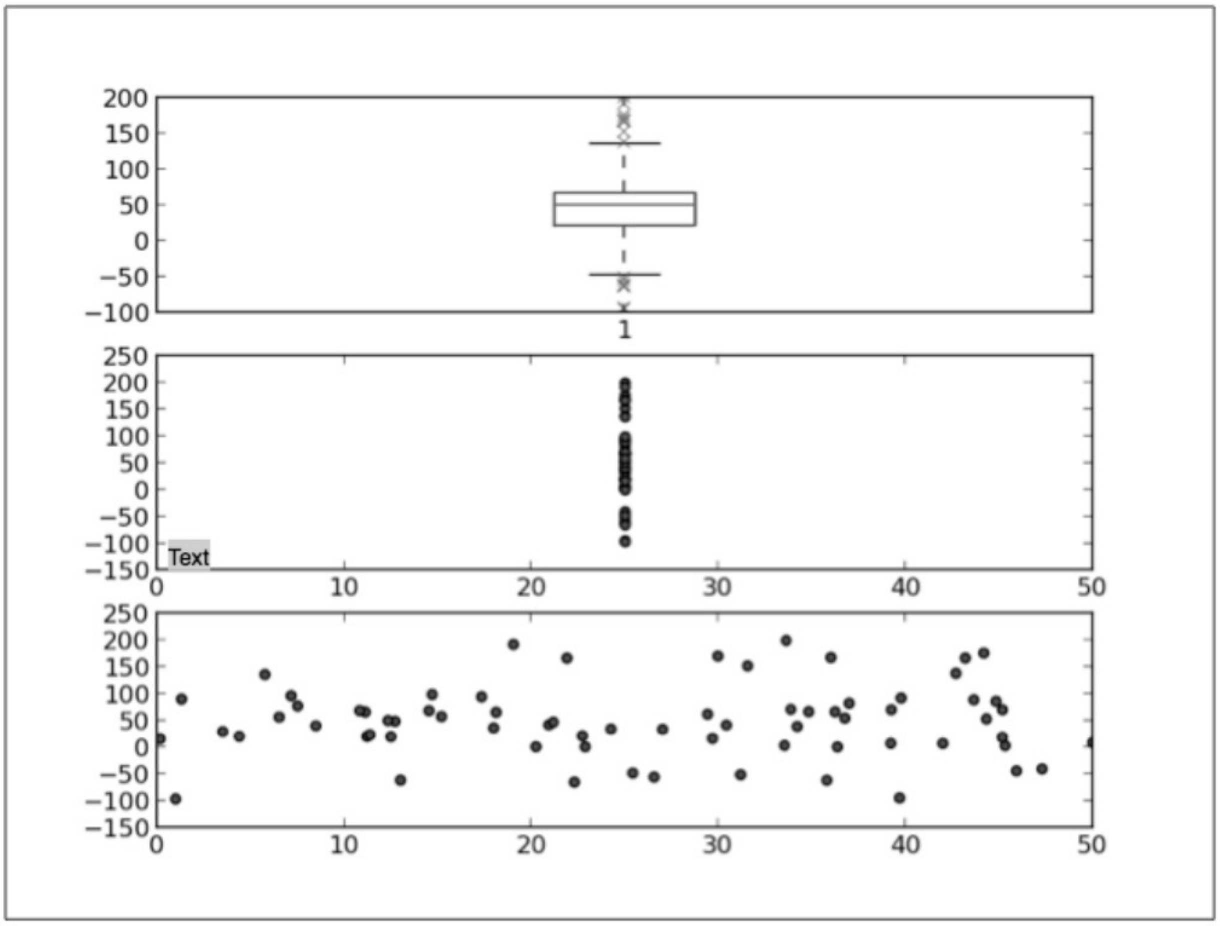
[illegible]whiskers[illegible]

[illegible]

```
from pylab import *
# fake up some data
spread= rand(50) * 100
center = ones(25) * 50
# generate some outliers high and low
flier_high = rand(10) * 100 + 100
flier_low = rand(10) * -100
# merge generated data set
data = concatenate((spread, center, flier_high, flier_low), 0)
subplot(311)
# basic plot
# 'gx' defining the outlier plotting properties
boxplot(data, 0, 'gx')
# compare this with similar scatter plot
subplot(312)
spread_1 = concatenate((spread, flier_high, flier_low), 0)
center_1 = ones(70) * 25
scatter(center_1, spread_1)
xlim([0, 50])
# and with another that is more appropriate for
# scatter plot
subplot(313)
center_2 = rand(70) * 50
```

```
scatter(center_2, spread_1)
xlim([0, 50])
show()
```

□□2-3□□□□□x□□□□□□□□□□



□2-3

□□□□□□□□□□□□□□□□□□□□□□□□X□□□□□25□□□□□□□□□□□□□□□□□□□□□□□□□□ inlier □□□□□□□outlier□□

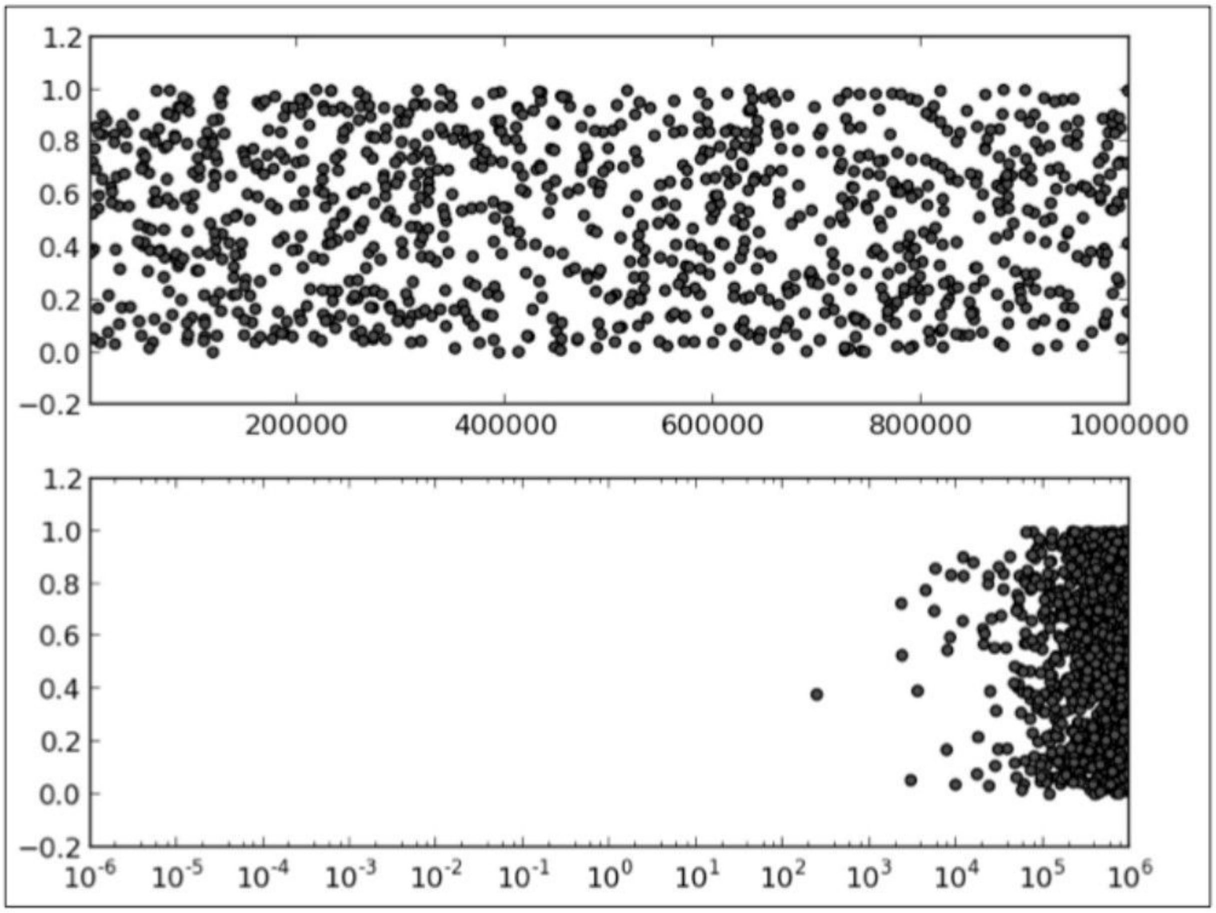
□□□□□□□X□□□□□□□□0□50□□□□□□□□□□□□□□□□□□□□□Y□□□□□□□□□□

□□□

```
# generate uniform data points
```

```
x = 1e6*rand(1000)
y = rand(1000)
figure()
# create first subplot
subplot(211)
# make scatter plot
scatter(x, y)
# limit x axis
xlim(1e-6, 1e6)
# create second subplot
subplot(212)
# make scatter plot
scatter(x,y)
# but make x axis logarithmic
xscale('log')
# set same x axis limit
xlim(1e-6, 1e6)
show()
```

□□2-4□□□□□□□□□



[illegible]2-4

[illegible]missing value[illegible] NumPy [illegible]

[illegible]OH[illegible] Ohio[illegible]OHIO[illegible]US-OH[illegible]OH-USA[illegible]Ohio[illegible] Microsoft Excel [illegible] OpenOffice.org Calc [illegible]Python[illegible]CSV[illegible]CSV[illegible]

[illegible]Python[illegible] readlines()[illegible]Python[illegible]

2.9.3 [illegible]

[illegible]OpenRefine[illegible]https://github.com/OpenRefine[illegible]“[illegible]”[illegible]

[illegible]

[illegible]statistical models[illegible]sampling theory[illegible]

2.10 [illegible]

Python[illegible]MB[illegible] 2GB [illegible]Python [illegible]

[illegible]

```
with open('/tmp/my_big_file', 'r') as bigfile:
    for line in bigfile:
        # line based operation, like 'print line'
```

[illegible]IO[illegible]seek()[illegible]tell()[illegible]read()[illegible]next()[illegible]C[illegible]

2.10.1 [illegible]

[illegible]1000[illegible]Python[illegible]

```
import sys
filename = sys.argv[1] # must pass valid file name
with open(filename, 'rb') as hugefile:
  chunksize = 1000
  readable = ''
  # if you want to stop after certain number of blocks
  # put condition in the while
  while hugefile:
    # if you want to start not from 1st byte
    # do a hugefile.seek(skipbytes) to skip
    # skipbytes of bytes from the file start
    start = hugefile.tell()
    print "starting at:", start
    file_block = '' # holds chunk_size of lines
    for _ in xrange(start, start + chunksize):
      line = hugefile.next()
      file_block = file_block + line
      print 'file_block', type(file_block), file_block
    readable = readable + file_block
    # tell where are we in file
    # file IO is usually buffered so tell()
    # will not be precise for every read.
    stop = hugefile.tell()
    print 'readable', type(readable), readable
    print 'reading bytes from %s to %s' % (start, stop)
    print 'read bytes total:', len(readable)
    # if you want to pause read between chucks
```

```
        # uncomment following line
        # raw_input()
```

[illegible]Python[illegible]

```
$ python ch02-chunk-read.py myhugefile.dat
```

2.10.2 [illegible]

[illegible]

[illegible] for [illegible]next()[illegible]file_block [illegible]readable[illegible]

[illegible]

while[illegible]raw_input()[illegible]

2.10.3 [illegible]

[illegible]Python[illegible]C [illegible]

[illegible]MapReduce[illegible]

[illegible]multiprocessing[illegible]Python[illegible]multiprocessing[illegible]threading[illegible]thread[illegible]

[illegible]

2.11 [illegible]

[illegible]

2.11.1 [illegible]

[illegible]
Python[illegible]

```
import time
import os
import sys
if len(sys.argv) != 2:
    print >> sys.stderr, "Please specify filename to read"
filename = sys.argv[1]
if not os.path.isfile(filename):
    print >> sys.stderr, "Given file: \"%s\" is not a file" %
filename
with open(filename,'r') as f:
    # Move to the end of file
    filesize = os.stat(filename)[6]
    f.seek(filesize)
    # endlessly loop
    while True:
        where = f.tell()
        # try reading a line
        line = f.readline()
        # if empty, go back
        if not line:
            time.sleep(1)
            f.seek(where)
        else:
            # , at the end prevents print to add newline, as
    readline()
            # already read that.
            print line,
```

2.11.2 [illegible]

[illegible] while True:[illegible] Ctrl+C [illegible] seek()[illegible]

[illegible]

2.11.3 [illegible]

[illegible]n[illegible] file.seek(filesize – N * avg_line_len) [illegible] avg_line_len[illegible]1024[illegible]readlines()[illegible][-N][illegible]

[illegible]HTTP[illegible]intermediate queue[illegible]

io[illegible]Python[illegible]2.6[illegible]io [illegible] Python 3.x [illegible]

[illegible]message queue[illegible] [illegible]message bus[illegible]

2.12 [illegible]NumPy[illegible]

[illegible]NumPy[illegible]SciPy[illegible]Python[illegible]

[illegible]n[illegible]
NumPy[illegible]

[illegible]3D[illegible]Z[illegible]4D[illegible]

[illegible]

[illegible]Python[illegible]

2.12.1 [illegible]

[illegible]SciPy[illegible]NumPy[illegible]SciPy[illegible]

```
$ sudo apt-get install python-scipy
```

[illegible]Windows[illegible]Python[illegible]EPD[illegible]1[illegible]“[illegible]”[illegible]

[illegible]

◆ BLAS [illegible]LAPACK[illegible]libblas [illegible] liblapack[illegible]

◆ C □ Fortran □□□□gcc □ gfortran□

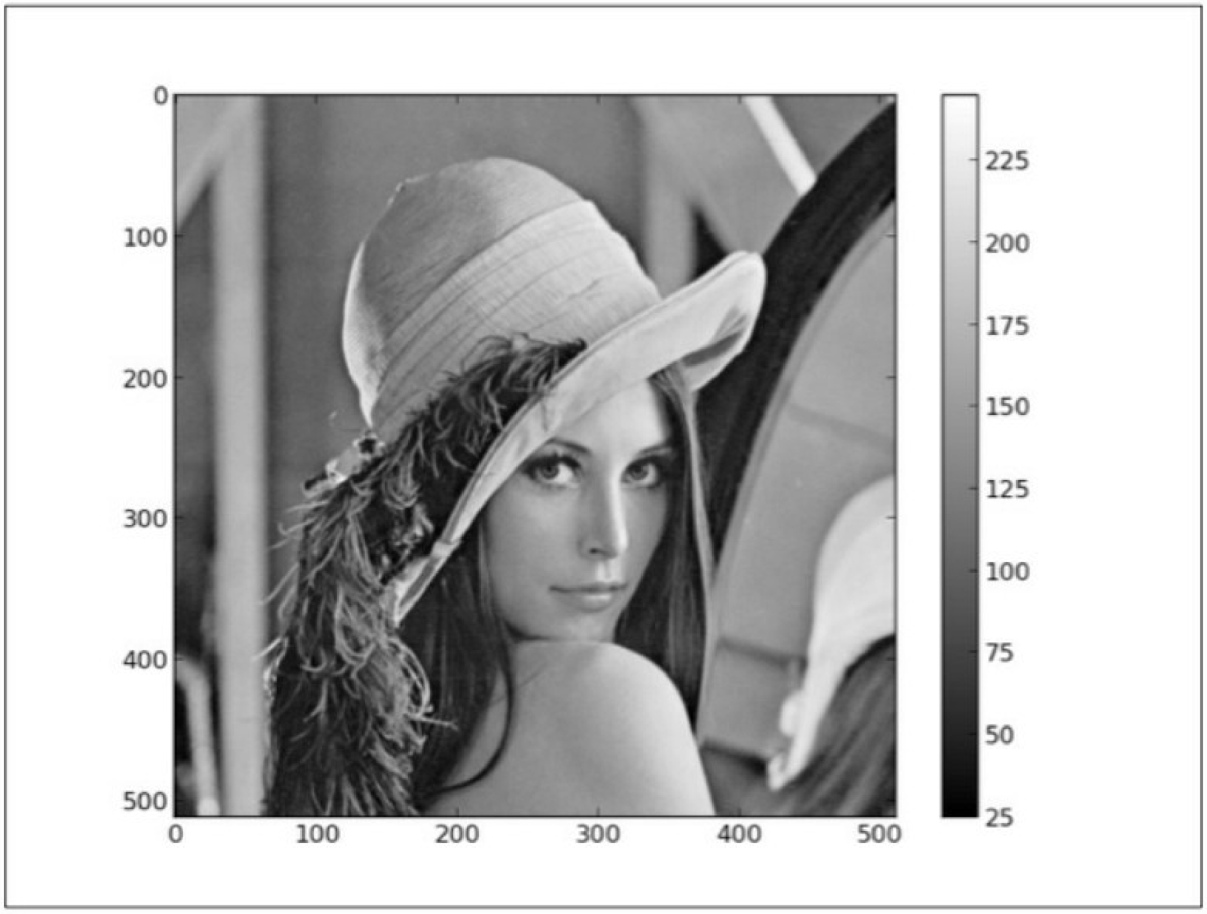
2.12.2 □□□□

□□Lena□□Lena□□□□□□□□□□□□□□□□□□□□□□

SciPy□□□□□□□□□□misc□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
import scipy.misc
import matplotlib.pyplot as plt
# load already prepared ndarray from scipy
lena = scipy.misc.lena()
# set the default colormap to gray
plt.gray()
plt.imshow(lena)
plt.colorbar()
plt.show()
```

□□□□□□□□□□□□□Lena□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□0——□□□255——□□□□□2-5□□□□



[illegible]2-5

[illegible]

```
print lena.shape
print lena.max()
print lena.dtype
```

[illegible]

```
(512, 512)
245
dtype('int32')
```

[illegible]

◆ 512 [illegible] 512 [illegible]

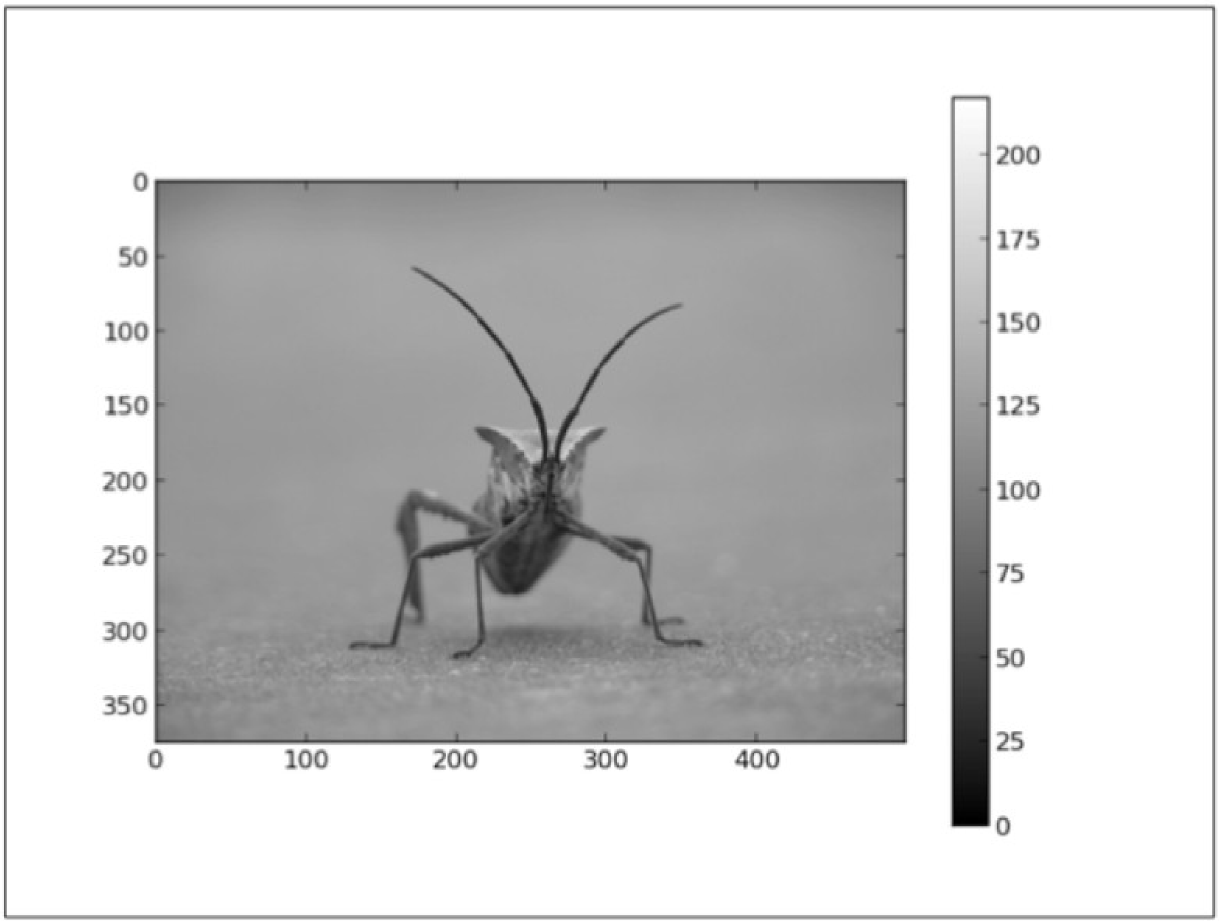
◆ [illegible] 245[2] [illegible]

◆ □□□□□□□□□□□□little endian□32 □□□□

□□□□ Python Image Library□PIL□□□□□□□□ 1 □“□□□□□□”□□□□□□□□□PIL□

```
import numpy
import Image
import matplotlib.pyplot as plt
bug = Image.open('stinkbug.png')
arr = numpy.array(bug.getdata(), numpy.uint8).reshape(bug.size[1],
bug.size[0], 3)
plt.gray()
plt.imshow(arr)
plt.colorbar()
plt.show()
```

□□□□□Lena□□□□□□□□□□□2-6□□□



[illegible]2-6

[illegible]PIL[illegible]

2.12.3 [illegible]

[illegible]Python[illegible]RGB[illegible]ndarray[illegible]NumPy[illegible]matplotlib[illegible]

```
import matplotlib.pyplot as plt
import scipy
import numpy
bug = scipy.misc.imread('stinkbug1.png')
# if you want to inspect the shape of the loaded image
# uncomment following line
#print bug.shape
# the original image is RGB having values for all three
# channels separately. We need to convert that to greyscale image
# by picking up just one channel.
# convert to gray
bug = bug[:,:,0]
```

bug[:,:,0][illegible]array slicing[illegible]NumPy [illegible]

```
>>> a = array(5, 1, 2, 3, 4)
>>> a[2:3]
array([2])
>>> a[:2]
array([5, 1])
>>> a[3:]
array([3, 4])
```

[illegible]

```
>>> b = array([[1,1,1],[2,2,2],[3,3,3]]) # matrix 3 x 3
>>> b[0,:] # pick first row
array([1,1,1])
>>> b[:,0] # we pick the first column
array([1,2,3])
```

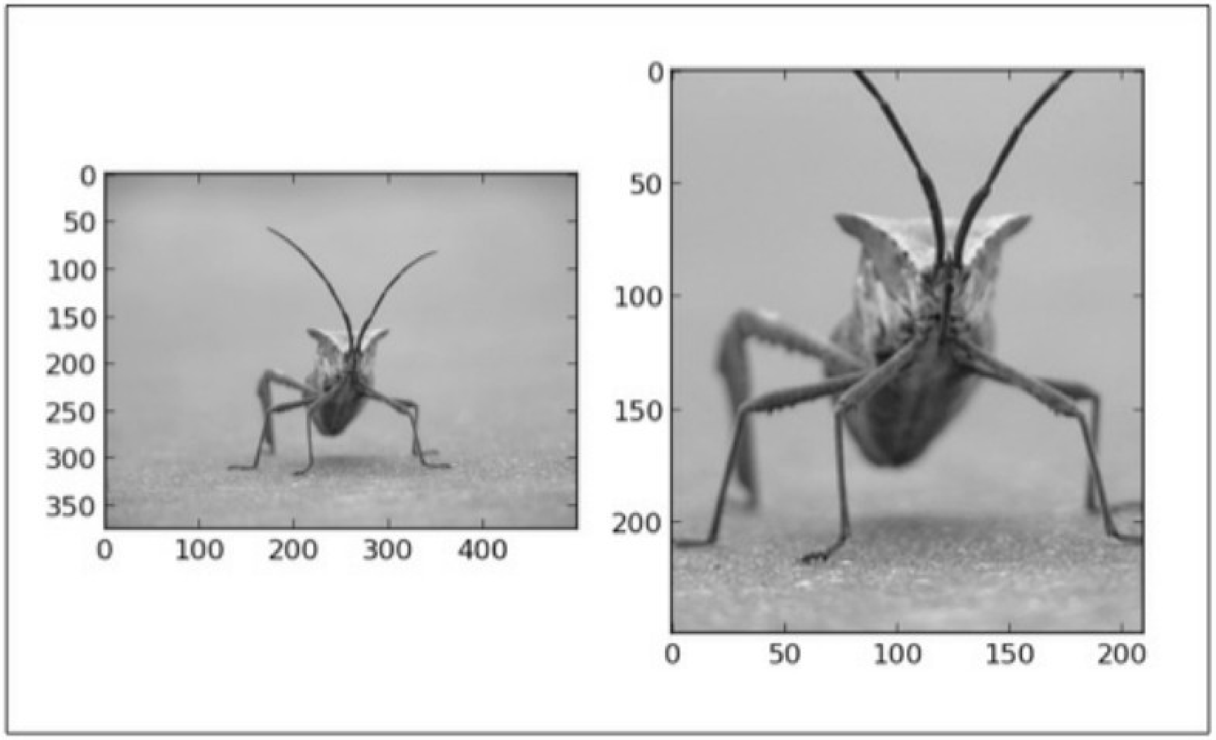
[illegible]

```
# show original image
plt.figure()
plt.gray()
plt.subplot(121)
plt.imshow(bug)
# show 'zoomed' region
zbug = bug[100:350,140:350]
```

□□□□□□□□□□□□□□□□□□□□□□□□NumPy□□□□□□□□□□□□□□□□□□□□□□□□□□□□□100□□250□□□140□□350□□□□□□□□□□□□□0□□□□□100□□□□101□□

```
plt.subplot(122)
plt.imshow(zbug)
plt.show()
```

□□□□□2-7□□□



□2-7

2.12.4 □□□□

[illegible] numpy.memmap [illegible]

```
import numpy
file_name = 'stinkbug.png'
image = numpy.memmap(file_name, dtype=numpy.uint8, shape = (375, 500))
```

[illegible]NumPy[illegible]NumPy[illegible]shape[illegible]file_name[illegible]Python [illegible] mmap[illegible]http://docs.python.org/2/ library/mmap.html[illegible]NumPy[illegible]memmap[illegible]Python[illegible]mmap [illegible]

[illegible] scikit-image [illegible]http://scikit-image.org/[illegible]NumPy/SciPy[illegible]scikit[illegible]scikit[illegible] http://scikit-image.org/docs/dev/auto_examples/[illegible]

2.13 [illegible]

[illegible]Python[illegible]NumPy/SciPy[illegible]

[illegible]

[illegible]Python[illegible]

[illegible]

2.13.1 [illegible]

[illegible]

◆ [illegible]Distribution or probability distribution[illegible]

◆ [illegible]Standard deviation[illegible]

◆ [illegible]Variance[illegible]

◆ [illegible]Population or statistical population[illegible]

◆ [illegible]Sample[illegible]

2.13.2 [illegible]

[illegible]Python[illegible]random[illegible]

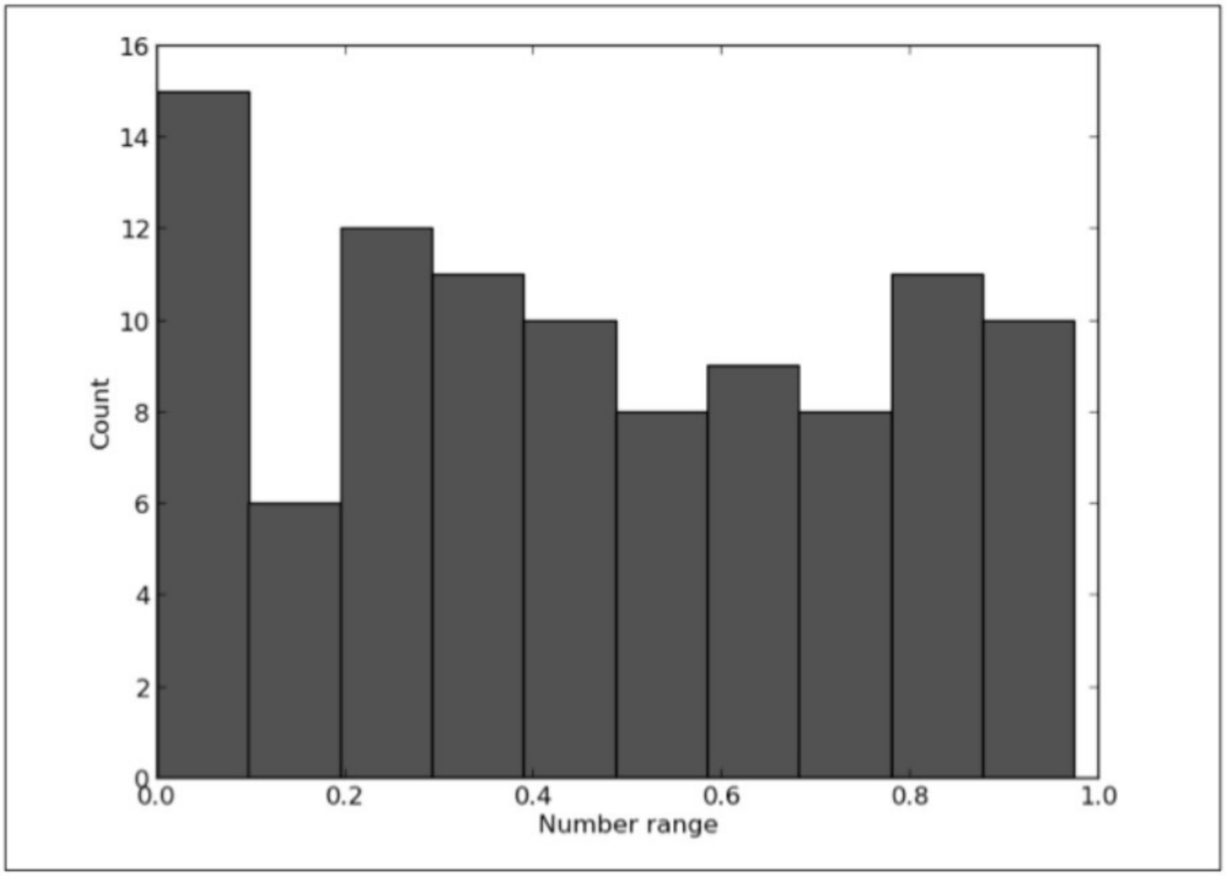
```
import pylab
import random
SAMPLE_SIZE = 100
# seed random generator
# if no argument provided
# uses system current time
random.seed()
# store generated random values here
```

```
real_rand_vars = []
# pick some random values
real_rand_vars = [random.random() for val in xrange(SIZE)]
# create histogram from data in 10 buckets
pylab.hist(real_rand_vars, 10)
# define x and y labels
pylab.xlabel("Number range")
pylab.ylabel("Count")
# show figure
pylab.show()
```

[illegible]2-8[illegible]

[illegible]SAMPLE_SIZE[illegible]10000[illegible]

[illegible] 0 [illegible] 1 [illegible] 1 [illegible] 6 [illegible] random.randint(min, max)[illegible] min [illegible] max [illegible] random.uniform(min, max)[illegible]



[illegible]2-8

[illegible]

```
import pylab
import random
# days to generate data for
duration = 100
# mean value
mean_inc = 0.2
# standard deviation
std_dev_inc = 1.2
# time series
x = range(duration)
y = []
```

```
price_today = 0
for i in x:
    next_delta = random.normalvariate(mean_inc, std_dev_inc)
    price_today += next_delta
    y.append(price_today)
pylab.plot(x,y)
pylab.xlabel("Time")
pylab.xlabel("Time")
pylab.ylabel("Value")
pylab.show()
```

[illegible]100[illegible]mean_inc[illegible]std_dev_inc[illegible]random.normalvariate()[illegible]price_today[illegible]

[illegible]

```
# coding: utf-8
import random
import matplotlib
import matplotlib.pyplot as plt
SAMPLE_SIZE = 1000
# histogram buckets
buckets = 100
plt.figure()
# we need to update font size just for this example
matplotlib.rcParams.update({'font.size': 7})
```

□□□□□□□□□□□□□□□□□□□□6×2□subplot□□□□□□□□□□□□□□□□□□[0,1)□□□□□□□□□normal distributed random variable□□

plt.subplot(621)

plt.xlabel("random.random")

# Return the next random floating point number in the range [0.0, 1.0).

res = [random.random() for _ in xrange(1, SAMPLE_SIZE)]

plt.hist(res, buckets)

□□□□□□□□□□□□□□□□□□□□□□□uniformlydistributedrandomvariable□□

plt.subplot(622)

plt.xlabel("random.uniform")

# Return a random floating point number N such that a <= N <= b for a

<= b and b <= N <= a for b < a.

# The end-point value b may or may not be included in the range

depending on floating-point rounding in the equation a + (b-a) *

random().

a= 1

b = SAMPLE_SIZE

res = [random.uniform(a, b) for _ in xrange(1, SAMPLE_SIZE)]

plt.hist□res,buckets□

□□□□□□□□□□□□triangular distribution□□

plt.subplot(623)

plt.xlabel("random.triangular")

# Return a random floating point number N such that low <= N <= high

and with the specified

# mode between those bounds. The low and high bounds default to zero and one. The mode

# argument defaults to the midpoint between the bounds, giving a

symmetric distribution.

low = 1

high = SAMPLE_SIZE

res = [random.triangular(low, high) for _ in xrange(1, SAMPLE_SIZE)]

plt.hist(res, buckets)

□□□□□ beta □□□beta distribution□□□□□□□ alpha □ beta □□□□ 0□□□□□0□1□□□

plt.subplot(624)

plt.xlabel("random.betavariate")

alpha = 1

beta = 10

res = [random.betavariate(alpha, beta) for _ in xrange(1, SAMPLE_SIZE)]

plt.hist(res, buckets)

□□□□□□□□□□□□exponential distribution□□lambd□□□1.0□□□□□□□□□□□□□□□□□□□□□lambda□□□Python□□□□□□□□□lambd□□□□□□□□□□□□□□□□□lambd□□□□□□□□□□□□□

plt.subplot(625)

plt.xlabel("random.expovariate")

lambd = 1.0 / ((SAMPLE_SIZE + 1) / 2.)

res = [random.expovariate(lambd) for _ in xrange(1, SAMPLE_SIZE)]

plt.hist(res, buckets)

□□□□□ gamma □□□gamma distribution□□□□□□ alpha □ beta □□□□□□□□□□□□□

$$PDF(x) = \frac{x^{a-1} e^{\frac{-x}{\beta}}}{\gamma(a)\beta^{a}}$$

□□□gamma□□□□□

plt.subplot(626)

plt.xlabel("random.gammavariate")

alpha = 1

beta = 10

res = [random.gammavariate(alpha, beta) for _ in xrange(1, SAMPLE_SIZE)]

plt.hist(res, buckets)

□□□□□□□□□□□Log normal distribution□□□□□□□□□□□□□□□□□□□□□mu□□□□□sigma□□□□□mu□□□□□□sigma□□□□□

plt.subplot(627)

plt.xlabel("random.lognormvariate")

mu = 1

sigma = 0.5

```
res = [random.lognormvariate(mu, sigma) for _ in xrange(1, SAMPLE_SIZE)]
plt.hist(res, buckets)
```

□□□□□□□□□□□□normal distribution□□□□□ mu□□□□□ sigma□

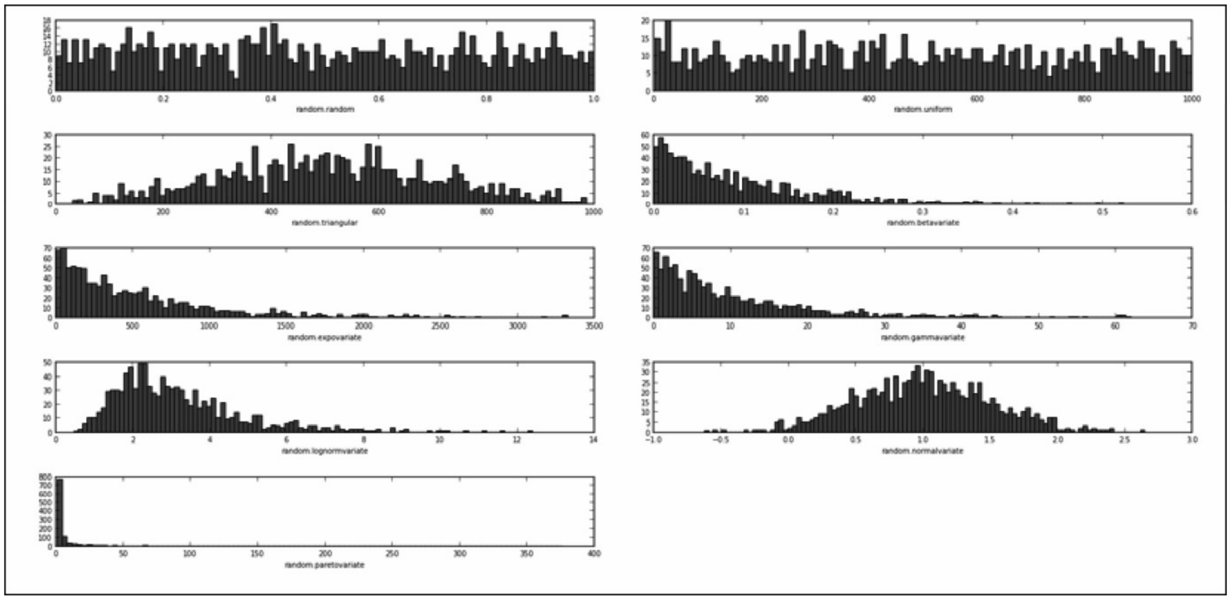
```
plt.subplot(628)
plt.xlabel("random.normalvariate")
mu = 1
sigma = 0.5
res = [random.normalvariate(mu, sigma) for _ in xrange(1, SAMPLE_SIZE)]
plt.hist(res, buckets)
```

□□□□□□□□□□□□Pareto distribution□□alpha □□□□□□

```
plt.subplot(629)
plt.xlabel("random.paretovariate")
alpha = 1
res = [random.paretovariate(alpha) for _ in xrange(1, SAMPLE_SIZE)]
plt.hist(res, buckets)
plt.tight_layout()
plt.show()
```

□□□□□□□□□□□□□□□□□□□□□□□□□□1000□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 9□□□□□2-9□□□□



[illegible]2-9

[illegible] seed()[illegible] random()[illegible]

[illegible]random.SystemRandom[illegible]os.urandom[illegible]os.urandom [illegible]entropy source[illegible]seed()[illegible]setstate()[illegible]

[illegible] Linux [illegible]/usr/share/dict/words[illegible][3]

```
import random
with open('/usr/share/dict/words', 'rt') as f:
   words = f.readlines()
words = [w.rstrip() for w in words]
for w in random.sample(words, 5):
   print w
```

[illegible]Unix[illegible]Windows[illegible]Mac[illegible]Windows[illegible]Project Gutenberg[illegible]Wiktionary[illegible]

British National Corpus [illegible] Dr Peter Norvig [illegible] http://norvig.com/big.txt[illegible]

2.14 [illegible]

[illegible]

2.14.1 [illegible]

[illegible]

[illegible]Python[illegible]NumPy[illegible]SciPy[illegible]matplotlib[illegible]

2.14.2 [illegible]

[illegible]rolling window[illegible]

[illegible]NumPy[illegible]convolve[illegible]NumPy[illegible]linspace[illegible]

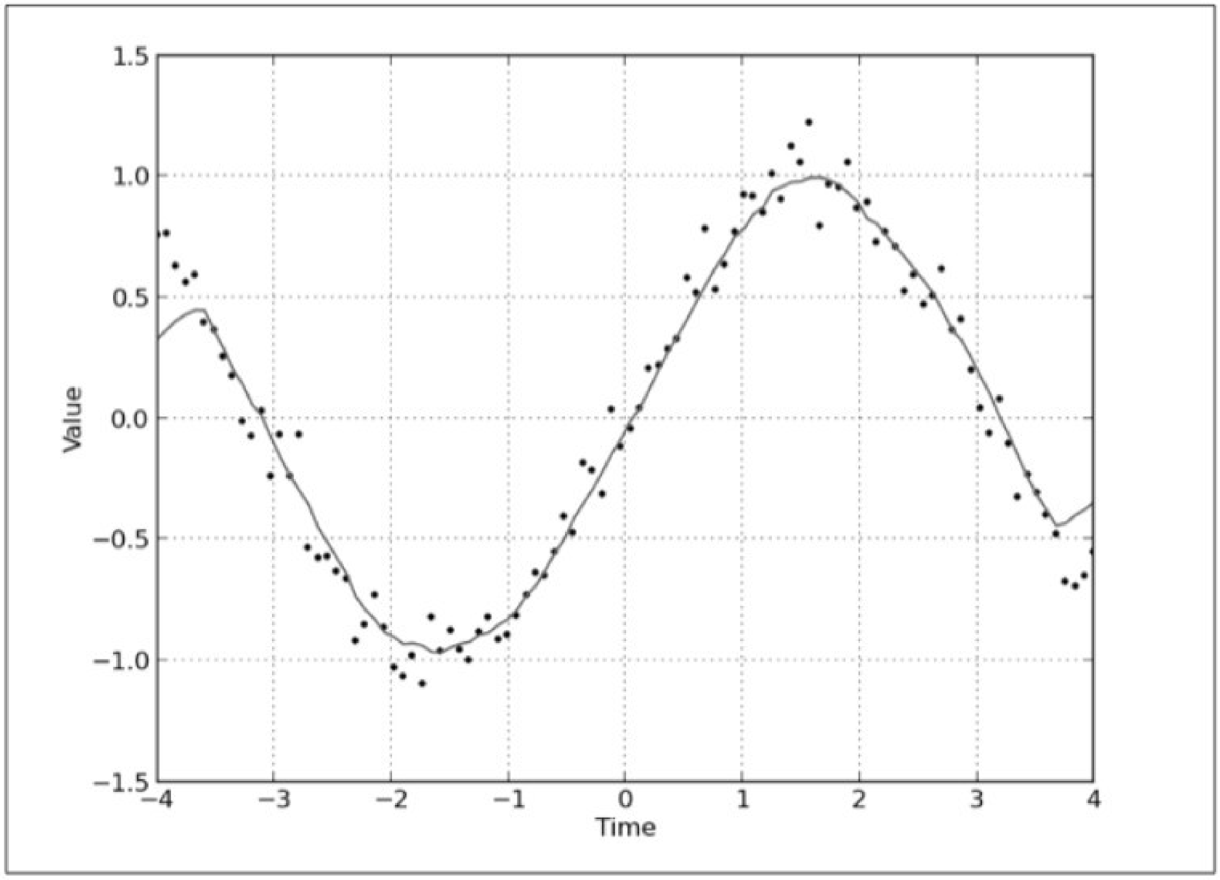
[illegible]ones[illegible]1[illegible]

2.14.3 [illegible]

[illegible]

```
from pylab import *
from numpy import *
def moving_average(interval, window_size):
    '''Compute convoluted window for given size
    '''
    window = ones(int(window_size)) / float(window_size)
    return convolve(interval, window, 'same')
t = linspace(-4, 4, 100)
y = sin(t) + randn(len(t))*0.1
plot(t, y, "k.")
# compute moving average
y_av = moving_average(y, 10)
plot(t, y_av,"r")
#xlim(0,1000)
xlabel("Time")
ylabel("Value")
grid(True)
show()
```

□□2-10□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□



□2-10

□□SciPy□□□□□□□□□□□□□□□□□□□

□□□SciPy Cookbook□□□□□□□□□□http://www.scipy.org/Cookbook/SignalSmooth□

import numpy

from numpy import *

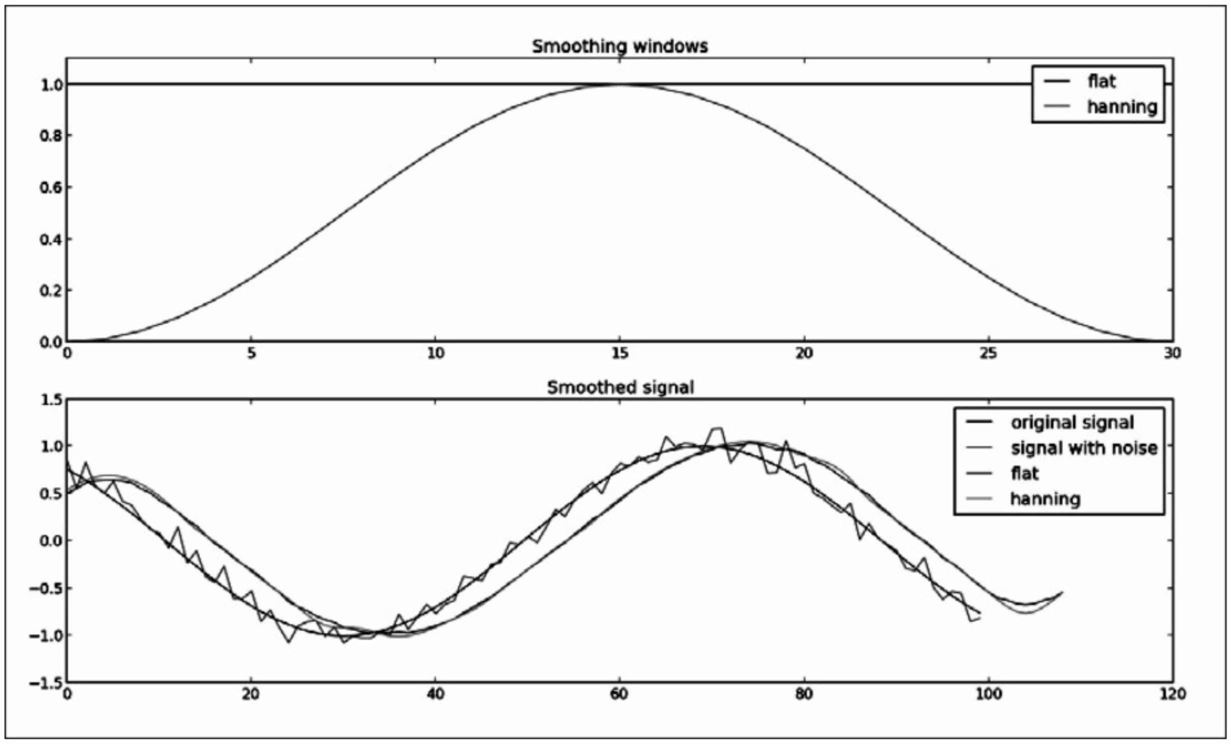
from pylab import *

# possible window type

WINDOWS = ['flat', 'hanning', 'hamming', 'bartlett', 'blackman']

```
    # if you want to see just two window type, comment
previous line,
    # and uncomment the following one
    # WINDOWS = ['flat', 'hanning']
    def smooth(x, window_len=11, window='hanning'):
      """
      Smooth the data using a window with requested size.
      Returns smoothed signal.
      x -- input signal
      window_len -- lenght of smoothing window
      window -- type of window: 'flat', 'hanning', 'hamming',
        'bartlett', 'blackman'
        flat window will produce a moving average
    smoothing.
      """
      if x.ndim != 1:
        raise ValueError, "smooth only accepts 1 dimension
    arrays."
      if x.size < window_len:
        raise ValueError, "Input vector needs to be bigger
    than window size."
      if window_len < 3:
        return x
      if not window in WINDOWS:
        raise ValueError("Window is one of 'flat', 'hanning',
    'hamming', "
          "'bartlett', 'blackman'")
      # adding reflected windows in front and at the end
      s=numpy.r_[x[window_len-1:0:-1],            x,            x[-1:-
  window_len:-1]]
      # pick windows type and do averaging
      if window == 'flat': #moving average
        w = numpy.ones(window_len, 'd')
      else:
        # call appropriate function in numpy
        w = eval('numpy.' + window + '(window_len)')
      #  NOTE:  length(output)  !=  length(input),  to  correct
  this:
      #  return  y[(window_len/2-1):-(window_len/2)]  instead
  of just y.
      y = numpy.convolve(w/w.sum(), s, mode='valid')
      return y
    #  Get  some  evenly  spaced  numbers  over  a  specified
interval.
    t = linspace(-4, 4, 100)
    # Make some noisy sinusoidal
    x = sin(t)
    xn = x + randn(len(t))*0.1
    # Smooth it
    y = smooth(x)
    # windows
    ws = 31
    subplot(211)
    plot(ones(ws))
    # draw on the same axes
    hold(True)
    # plot for every windows
    for w in WINDOWS[1:]:
      eval('plot('+w+'(ws) )')
    # configure axis properties
    axis([0, 30, 0, 1.1])
    # add legend for every window
    legend(WINDOWS)
    title("Smoothing windows")
    # add second plot
    subplot(212)
    # draw original signal
    plot(x)
    # and signal with added noise
    plot(xn)
    # smooth signal with noise for every possible windowing
algorithm
    for w in WINDOWS:
      plot(smooth(xn, 10, w))
    # add legend for every graph
    l=['original signal', 'signal with noise']
    l.extend(WINDOWS)
    legend(l)
    title("Smoothed signal")
    show()
```

[illegible]2-11[illegible]



[illegible]2-11

2.14.4 [illegible]

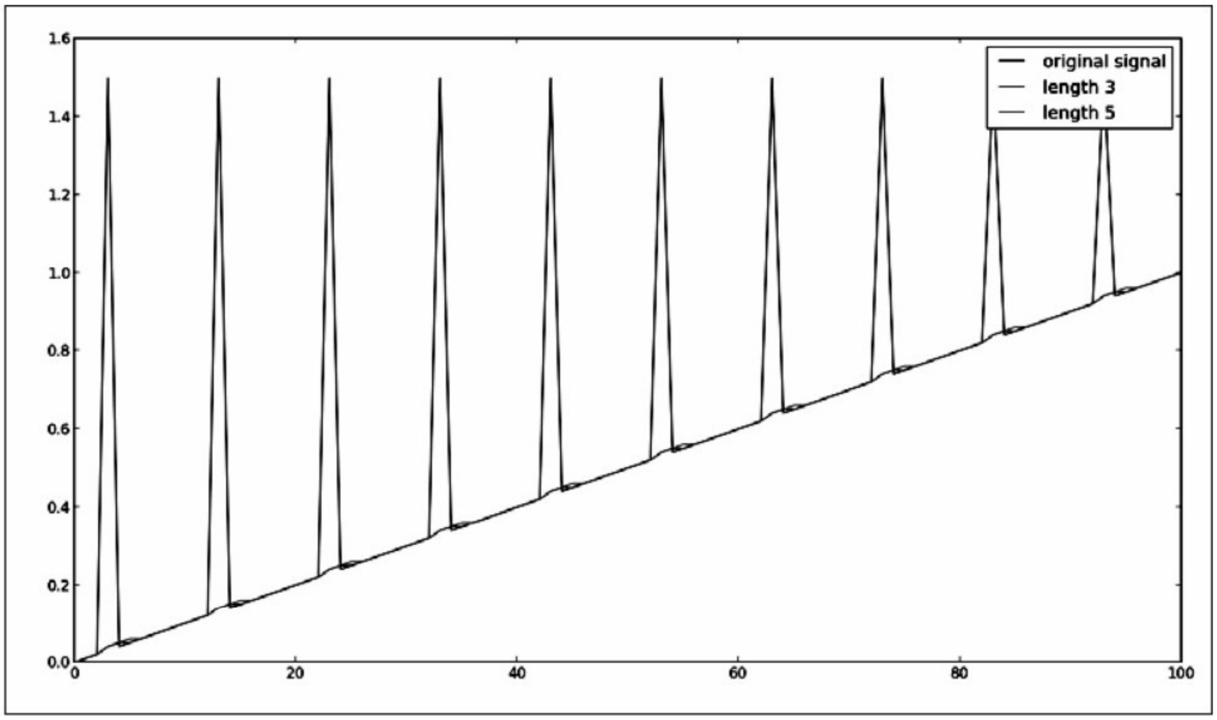
[illegible]Median Filter[illegible]

[illegible]SciPy[illegible]

```
import numpy as np
import pylab as p
import scipy.signal as signal
# get some linear data
```

```
x = np.linspace (0, 1, 101)
# add some noisy signal
x[3::10] = 1.5
p.plot(x)
p.plot(signal.medfilt(x,3))
p.plot(signal.medfilt(x,5))
p.legend(['original signal', 'length 3','length 5'])
p.show ()
```

[illegible]2-12[illegible]



[illegible]2-12

[illegible]

[illegible]“[illegible]”[illegible]least-squares curve fitting[illegible]

[illegible]

[1]. outlier[illegible]

[2]. [illegible] 254[illegible]

[3]. [illegible]/usr/share/dicts/words

□3□ □□□□□□□□

□□□□□□□□□□□□matplotlib□□□□□□□□□□□□

◆ □□□□□□——□□□□□□□□□□□□□

◆ □□□□□□□□□□□□

◆ □□□□□□□□□

◆ □□□□□□□□□□□□□□□□□

◆ □□□□□□□□□□□□

◆ □□□□□□□

◆ □□□□□□□□□

◆ □□□□□

◆ □□□□□□□

◆ □□□□

◆ □□□□□□□□□□

◆ □□□□□□□□□□□

3.1 □□

□□□□□□□matplotlib□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□matplotlib□□

Matplotlib□□□□□□□□□□□□□□□□□□□2D□□□3D□□□□□□□□□□□□□matplotlib□□□

[illegible]

3.2 [illegible]——[illegible]

[illegible]

3.2.1 [illegible]

[illegible] matplotlib.pyplot [illegible]

3.2.2 [illegible]

[illegible]IPython[illegible]IPython[illegible]

1.[illegible]IPython[illegible]

```
$ ipython --pylab
```

2.[illegible] matplotlib plot [illegible]

```
In [1]: plot([1,2,3,2,3,2,2,1])
Out[1]: [<matplotlib.lines.Line2D at 0x412fb50>]
```

[illegible]3-1[illegible]

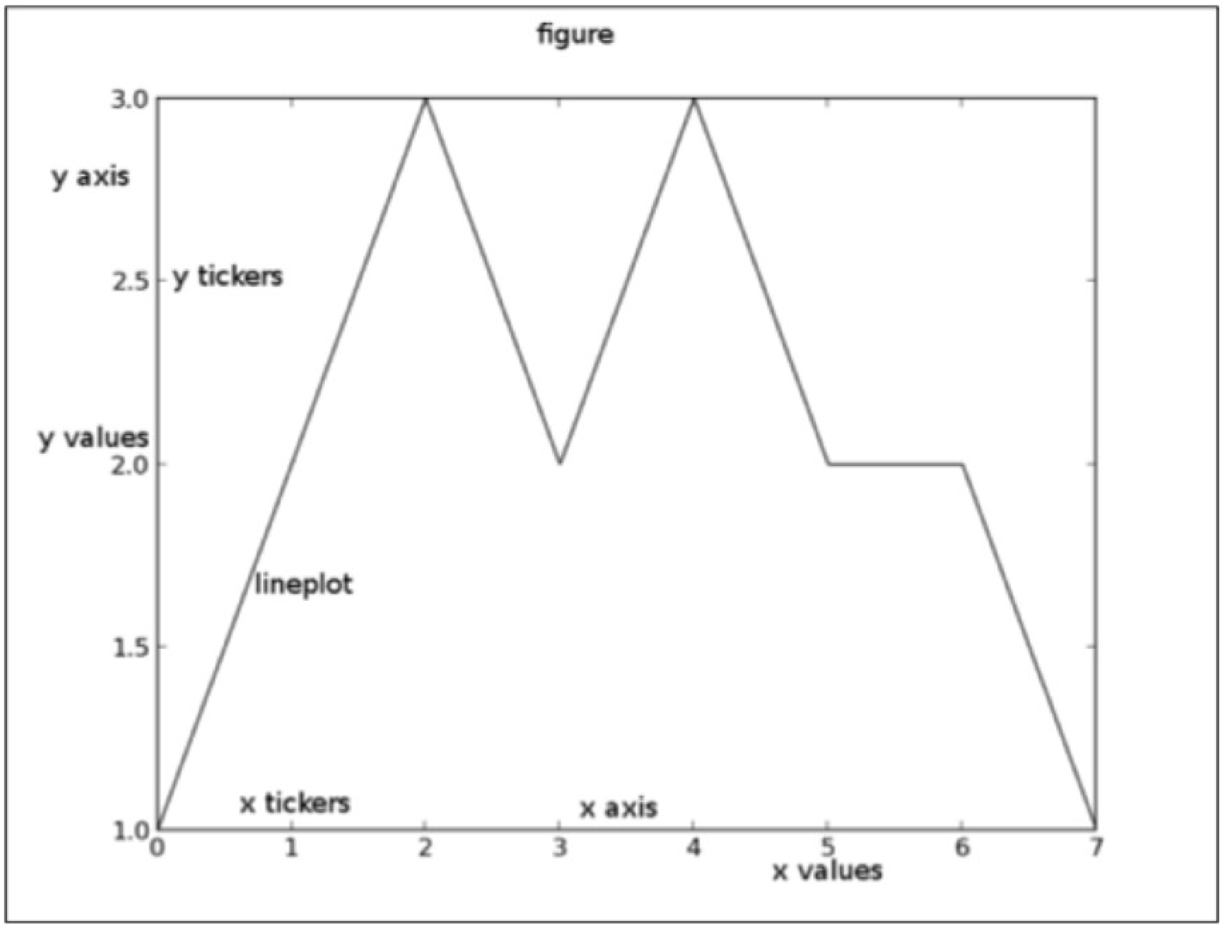
Matplotlib[illegible]

◆ x [illegible] y [illegible]

◆ x [illegible] y [illegible]

◆ x [illegible] y [illegible]

◆ [illegible]



□3-1

□□□□□□□□□plot()□□□y□□□□plot()□x□□□□□□□□□□□0□7□□□□□y□□□□1□□

□□□□□□plot()□□□□□□□x□□□□□□□IPython□□□□□□□□□

In [2]: plot([4,3,2,1],[1,2,3,4])

Out[2]: [<matplotlib.lines.Line2D at 0x31444d0>]

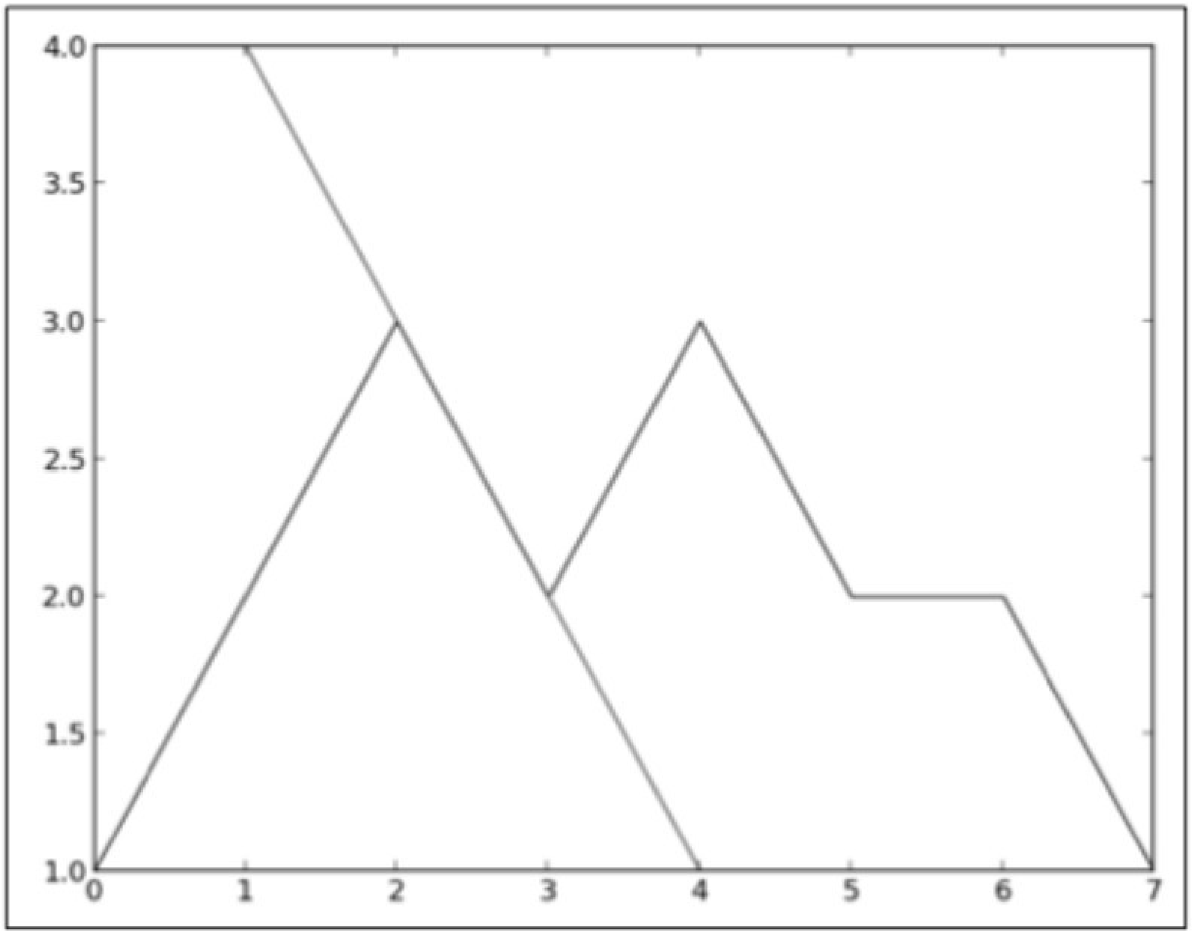
□□IPython□□□□□□□□□□□□□□□In[2]□Out[2])□□□□□□□□□□□□□□□□□□□□□□IPython□□□□□□□□□□□□□□□□□□□Python□□□□□□

□□□□□□IPython□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□

□□□□□□□3-2□□□□□□

□□□□□matplotlib□□□□y□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□

□□□□□hold□□□□□□□hold(False)□□□□□□□□□□□□□□□□□□□□□□□
□□□□IPython□pylab□□□□□□□□□□□□□□□Python□□□□hold□□□□□□□
□[1]□



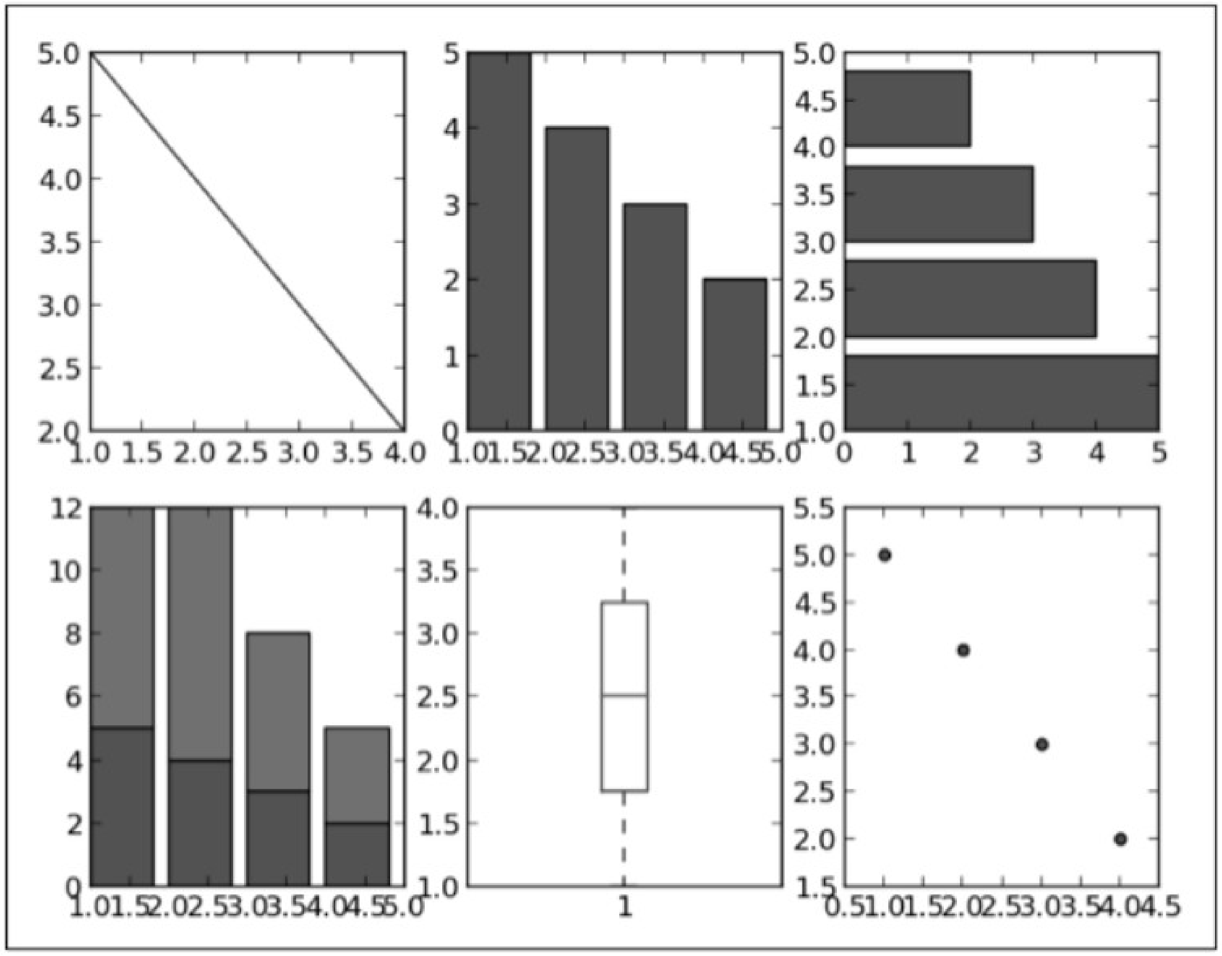
□3-2

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ IPython □□□□□□□
□□□□□□□□□□Python□□□□□□□

```
from matplotlib.pyplot import *
# some simple data
x = [1,2,3,4]
y = [5,4,3,2]
# create new figure
figure()
# divide subplots into 2 x 3 grid
# and select #1
subplot(231)
plot(x, y)
# select #2
subplot(232)
bar(x, y)
   # horizontal bar-charts
subplot(233)
barh(x, y)
# create stacked bar charts
subplot(234)
bar(x, y)
# we need more data for stacked bar charts
y1 = [7,8,5,3]
bar(x, y1, bottom=y, color = 'r')
# box plot
subplot(235)
boxplot(x)
# scatter plot
subplot(236)
```

```
scatter(x,y)
show()
```

[illegible]3-3[illegible]



[illegible]3-3

3.2.3 [illegible]

[illegible]figure()[illegible] sample charts[illegible] figure()[illegible]

[illegible] subplot(231) [illegible] 2 × 3 [illegible] subplot(3,2,1)[illegible]

□□

□□□□□□□□□□□□□□□□□□bar()□□□□□□□□barh()□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□bottom=y□□□□□□□□□□□□□□□□□□□□
□□□□

□□□□ boxplot()□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□

□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

3.2.4 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□5□□□□

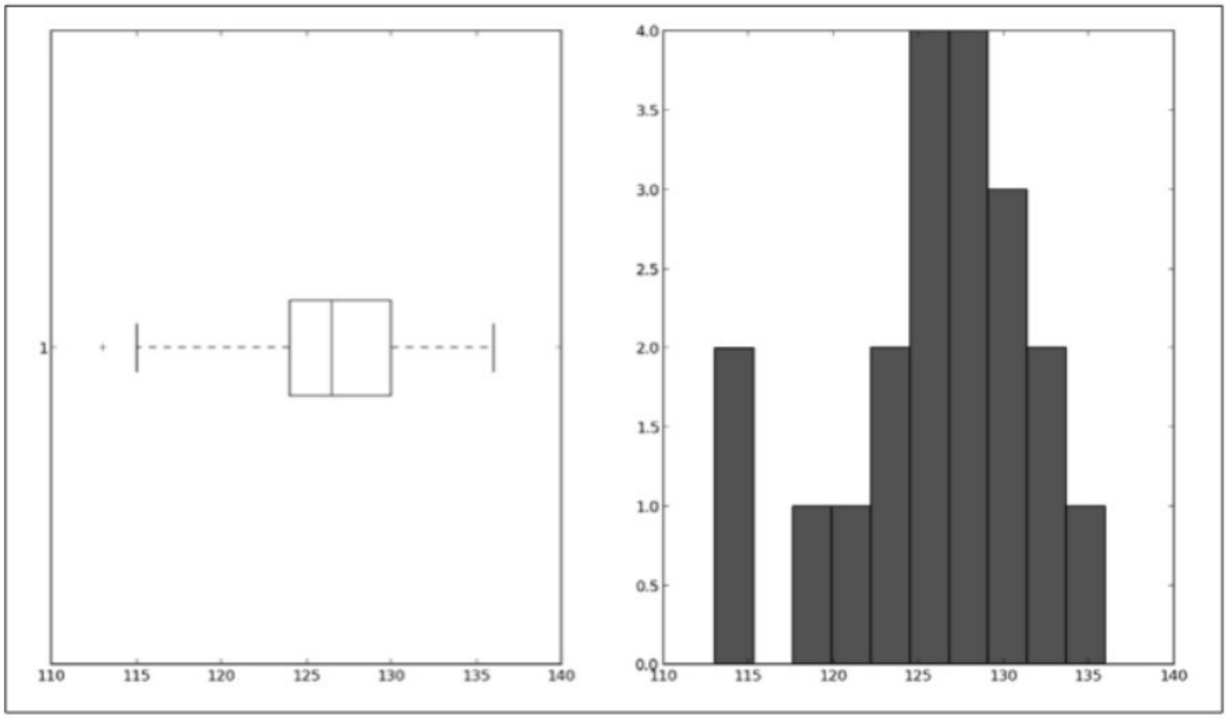
- ◆ □□□□□□□□□□□□□
- ◆ □□□□□□□□□□□□□□□□□□□ 25%□□□
- ◆ □□□□□□□□□□□
- ◆ □□□□□□□□□□□□□□□□□□□ 25%□□□
- ◆ □□□□□□□□□□□□□□□

□□

```
from pylab import *
dataset = [113, 115, 119, 121, 124,
   124, 125, 126, 126, 126,
   127, 127, 128, 129, 130,
   130, 131, 132, 133, 136]
subplot(121)
boxplot(dataset, vert=False)
```

```
subplot(122)
hist(dataset)
show()
```

□□□□□□□3-4□□□



□3-4

□□

3.3 □□□□□□□□□□

□□

3.3.1 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□(x,y)□□□

3.3.2 □□□□

□□□□□□□-Pi[2] □Pi□□□□□□□□□□□□256□□□□□□□□□□□□□□□□sin(x)□□cos(x)□□□□□□□□□□□□

```
import matplotlib.pyplot as pl
import numpy as np
x = np.linspace(-np.pi, np.pi, 256, endpoint=True)
y = np.cos(x)
y1 = np.sin(x)
pl.plot(x,y)
pl.plot(x,y1)
pl.show()
```

□□□□□□□3-5□□□

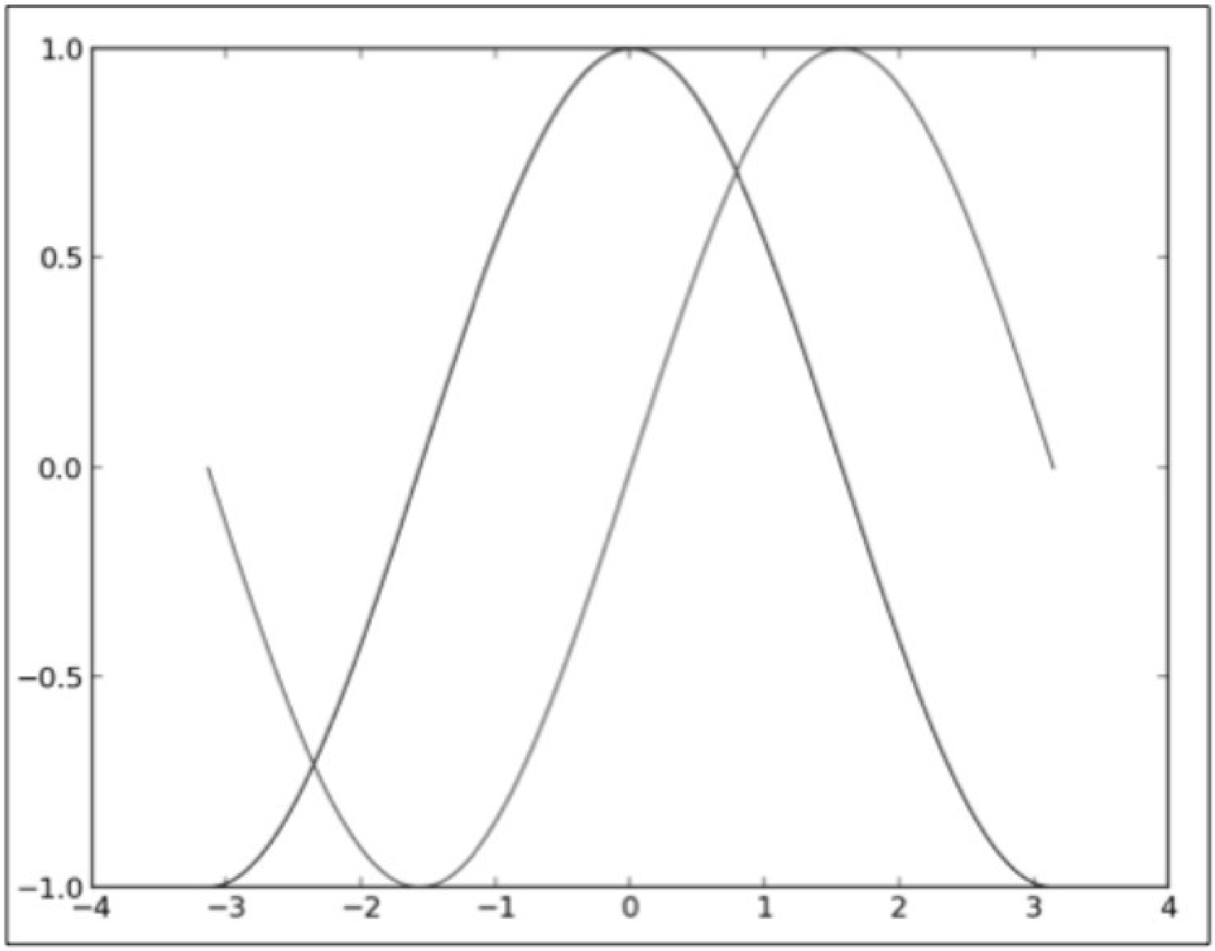


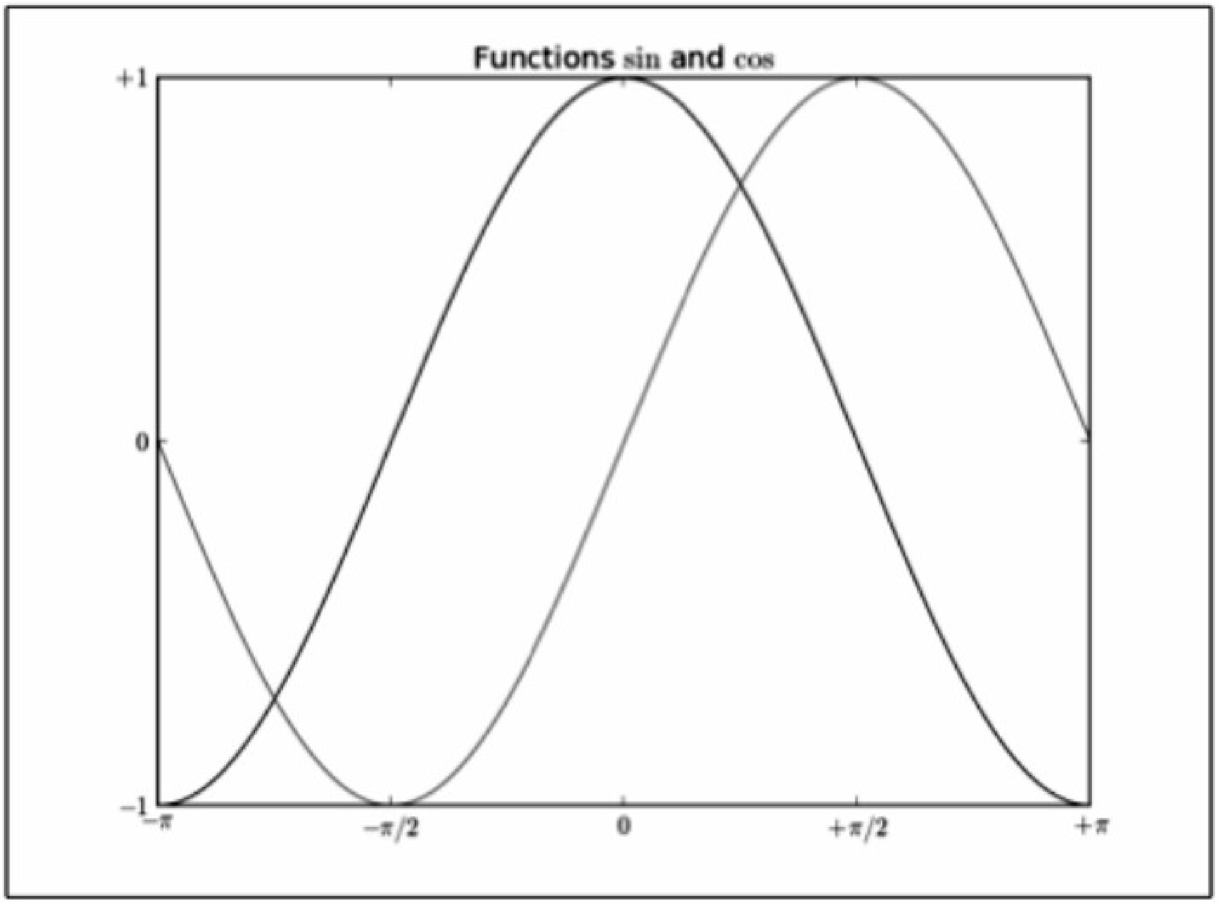
图3-5

[illegible]

```
from pylab import *
import numpy as np
# generate uniformly distributed
# 256 points from -pi to pi, inclusive
x = np.linspace(-np.pi, np.pi, 256, endpoint=True)
# these are vectorised versions
# of math.cos, and math.sin in built-in Python maths
# compute cos for every x
y = np.cos(x)
# compute sin for every x
```

```
y1 = np.sin(x)
# plot cos
plot(x, y)
# plot sin
plot(x, y1)
# define plot title
title("Functions $\sin$ and $\cos$")
# set x limit
xlim(-3.0, 3.0)
# set y limit
ylim(-1.0, 1.0)
# format ticks at specific values
xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
   [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
yticks([-1, 0, +1],
   [r'$-1$', r'$0$', r'$+1$'])
show()
```

[illegible]3-6[illegible]



[illegible]3-6

[illegible]$\sin$[illegible]$-\pi$[illegible]LaTex[illegible]

3.4 [illegible]

[illegible]matplotlib[illegible]

3.4.1 [illegible]

[illegible]IPython[illegible]

```
$ ipython --pylab
```

3.4.2 □□□□

□□□□□□□□□□□□□□□□□□□□□ axis()□□□□□□□□□□

```
In [1]: axis()
Out[1]: (0.0, 1.0, 0.0, 1.0)
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□xmin□xmax□ymin□ymax□□□□□□□□x□□y□□□

```
In [2]: l = [-1, 1, -10, 10]
In [3]: axis(l)
Out[3]: [-1, 1, -10, 10]
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□**kwargs□□□□□□□□□□□□xmax□□□□□

3.4.3 □□□□

□□□□axis()□□□□□□□matplotlib□□□□□□□□□□□□□□□□□□□□□□□□□□ axis()□□□□□□□□□□□□matplotlib□□autoscale()(matplotlib.pyplot.autoscale())[3]□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□ matplotlib.pyplot.axes()□□□□□□□□□□□□□□□□□□rect□□□□□□0□1□□□left□bottom□width□height□□□□□axisbg□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□sharex/sharey□□□□□□□□□□□□□□□□□x/y□□□□□□□□□ polar □□□□□□□□□□□□polar axes□□

□□

□□□□□□□□□□□□□□□□□□□□matplotlib.pyploy.axhline()□□matplotlib.pyplot.axvline()□axhline()□axvline()□□□□□□□□□x□y□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□axhline()□□□□□□□□□□□y □□□□xmin □ xmax□axvline()□□□□□□□□□□ x□□□□ymin□ymax□

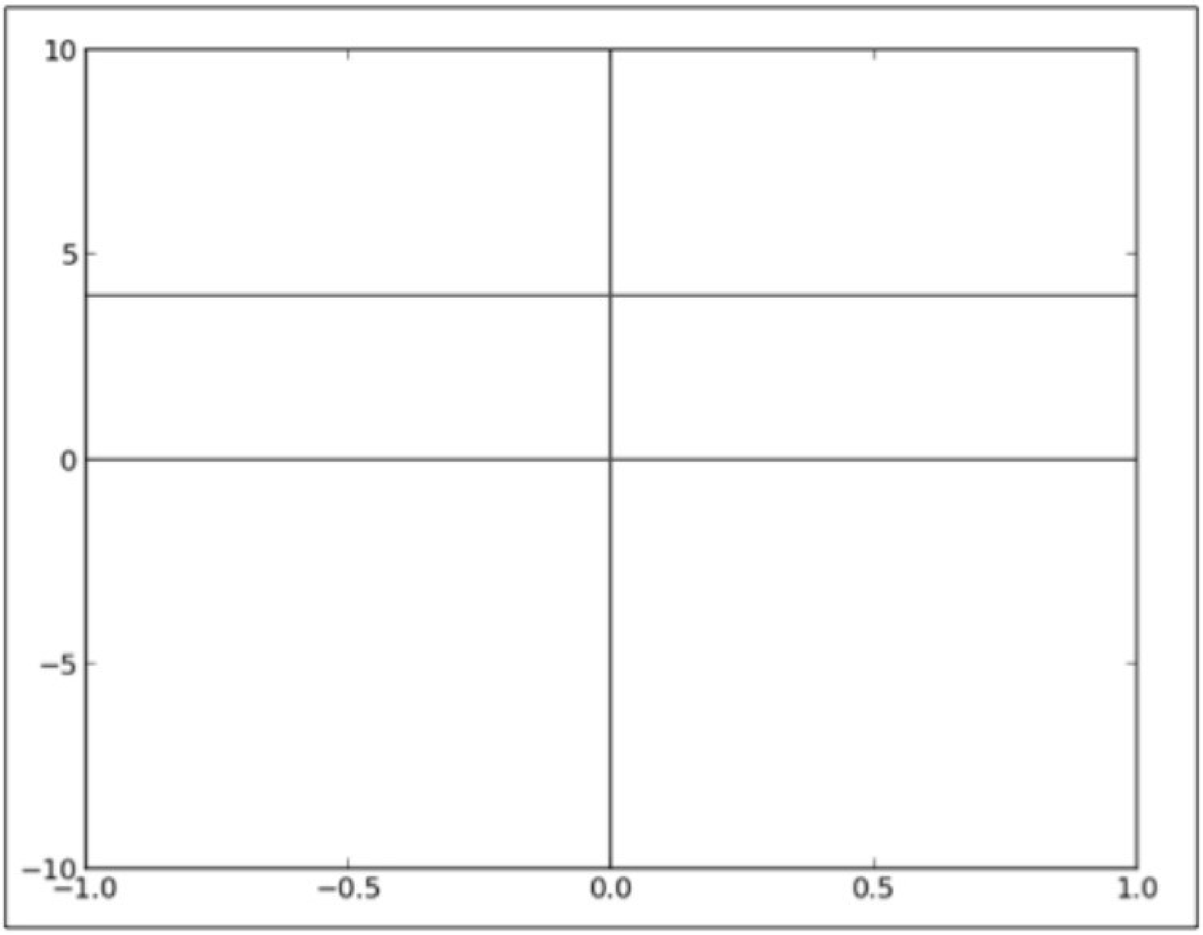
□□□□□□□□□□□□□□□□□IPython□□□□□□

```
In [3]: axhline()
Out[3]: <matplotlib.lines.Line2D at 0x414ecd0>
In [4]: axvline()
Out[4]: <matplotlib.lines.Line2D at 0x4152490>
In [5]: axhline(4)
Out[5]: <matplotlib.lines.Line2D at 0x4152850>
```

□□□□3-7□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□axhline()□□□□□□□y=0□□□□□axvline()□□□□□x=0□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□matplotlib.pyplot.axhspan()□matplotlib.pyplot.axvspan()[4]□axhspan()□□□□□ymin□ymax□□□□□□□□□□□□□□□axvspan()□□□□□xmin□xmax□□□□□□□□□□□□



[illegible]3-7

3.4.4 [illegible]

[illegible] matplotlib.pyplot.grid()[illegible]

◆ which[illegible]major[illegible]minor [illegible] both[illegible]

◆ axis[illegible]both[illegible]x [illegible] y[illegible]

[illegible] matplotlib.pyplot.axis()[illegible]Python[illegible]matplotlib.axes.Axes[illegible]matplotlib.axis.Axis [illegible] matplotlib.axis. XAxis [illegible] x [illegible] matplotlib.axis.YAxis[illegible]y[illegible]

[illegible]
matplot.pyplot [illegible]

3.5 [illegible]

[illegible]

3.5.1 [illegible]

[illegible]

[illegible]Colorbrewer2[illegible]http://colorbrewer2.org/[illegible]

[illegible]

3.5.2 [illegible]

[illegible]

[illegible]plot()[illegible]

plot(x, y, linewidth=1.5)

[illegible] plot()[illegible]matplotlib.lines.Line2D[illegible]setter[illegible]

line, = plot(x, y)

line.set_linewidth(1.5)

[illegible]MATLAB©[illegible]——[illegible]setp()[illegible]

```
lines = plot(x, y)
setp(lines, 'linewidth', 1.5)
```

[illegible]setp[illegible]

```
setp(lines, linewidth=1.5)
```

[illegible]

3.5.3 [illegible]

[illegible]matplotlib.lines.Line2D[illegible]3-1[illegible]

[illegible]3-1

属　性	类　型	描　述
alpha	浮点值	alpha 值用来设置混色，并不是所有后端都支持
color 或 c	任意 matplotlib 颜色	设置线条颜色
dashes	以点为单位的 on/off 序列[①]	设置破折号序列，如果 seq 为空或者如果 seq=[None, None]，linestyle 将被设置为 solid
label	任意字符串	为图例设置标签值
linestyle 或 ls	['-' | '--' | '-.' | ':' | 'steps' | ...]	设置线条风格（也接受 drawstyles 的值）
linewidth 或 lw	以点为单位的浮点值	设置以点为单位的线宽
marker	[7 | 4 | 5 | 6 | 'o' | 'D' | 'h' | 'H' | '_' | '' | 'None' | ' ' | None | '8' | 'p' | ',' | '+' | '.' | 's' | '*' | 'd' | 3 | 0 | 1 | 2 | '1' | '3' | '4' | '2' | 'v' | '<' | '>' | '^' | '|' | 'x' | '$...$' | tuple | Nx2 array]	设置线条标记
markeredgecolor 或 mec	任意 matplotlib 颜色	设置标记的边缘颜色
markeredgewidth 或 mew	以点为单位的浮点值	设置以点为单位的标记边缘宽度
markerfacecolor 或 mfc	任意 matplotlib 颜色	设置标记的颜色
markersize 或 ms	浮点值	设置以点为单位的标记大小
solid_capstyle	['butt' | 'round' | 'projecting']	设置实线的线端风格
solid_joinstyle	['miter' | 'round' | 'bevel']	设置实线的连接风格
visible	[True | False]	显示或隐藏 artist
xdata	np.array	设置 *x* 的 np.array 值

□□□□□□□□□□□□□□□□□□□□ dashes □□□[1,5,10]□□□□□□□□ 1 □□□□□□□□□□□□□□5□□□□□□□□□□□□□10□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□

属　性	类　型	描　述
ydata	np.array	设置 y 的 np.array 值
Zorder	任意数字	为 artist 设置 z 轴顺序，低 Zorder 的 artist 会先绘制 如果在屏幕上 x 轴水平向右，y 轴垂直向上，那么 z 轴将指向观察者。这样，0 表示在屏幕上，1 表示上面的一层，以此类推。

□□□□□□□□□□□□□

线条风格	描　述	线条风格	描　述
'-'	实线	':'	虚线
'--'	破折线	'None', ' ', ''	什么都不画
'-.'	点划线		

□□□□□□□□□□□□□□□□

标　记	描　述	标　记	描　述
'o'	圆圈	'.'	点
'D'	菱形	's'	正方形
'h'	六边形 1	'*'	星号
'H'	六边形 2	'd'	小菱形
'_'	水平线	'v'	一角朝下的三角形
'','None',' ',None	无	'<'	一角朝左的三角形
'8'	八边形	'>'	一角朝右的三角形
'p'	五边形	'^'	一角朝上的三角形
','	像素	'|'	竖线
'+'	加号	'x'	X

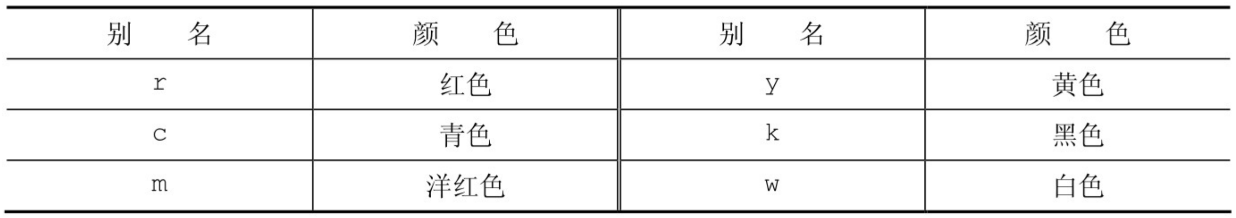
□□

□□□□□□matplotlib.pyplot.colors()□□matplotlib□□□□□□□□□□3-2□□□

□3-2

别　名	颜　色	别　名	颜　色
b	蓝色	g	绿色

□□



别　名	颜　色	别　名	颜　色
r	红色	y	黄色
c	青色	k	黑色
m	洋红色	w	白色

[illegible]matplotlib[illegible]

[illegible]——[illegible]——[illegible]HTML[illegible]

color = '#eeefff'

[illegible] HTML [illegible]'red', 'chartreuse'[illegible][0, 1][illegible] RGB [illegible]

color = (0.3, 0.3, 0.4)

[illegible]title()[illegible]

title('Title in a custom color', color='#123456')

[illegible]

[illegible] matplotlib.pyplot.axes() [illegible] matplotlib.pyplot.subplot()[illegible]axisbg[illegible]

subplot(111, axisbg=(0.1843, 0.3098, 0.3098))

3.6 [illegible]

[illegible]

3.6.1 [illegible]

[illegible]figure[illegible][5][illegible]subplots[illegible]

□ matplotlib □□□□ figure()□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ plot()□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ plot□□□□□□□□□subplot()□□□□□□□□□□□□plot□□□□□□□□□□□□□plot□□□□

□□□□□□□□□□□□□□□□□□matplotlib.axes.Axes□□□□□□□□□□□□□□□plot□□□□□□□□□□□□□□□□□□□□□□□□□plot□□□□□□plot□□

3.6.2 □□□□

□□□□□□□□□□□□□□□□□□tick locator□——□□□□□□□□□□——□□□□□□□tick formatter□——□□□□□□□□□□——□□□□□□□□□□□major ticks□□□□□□minor ticks□□□

□□□□□□□ matplotlib.pyplot.locator_params()□□ plot□□□□□□□□□□□□□□tight view□□

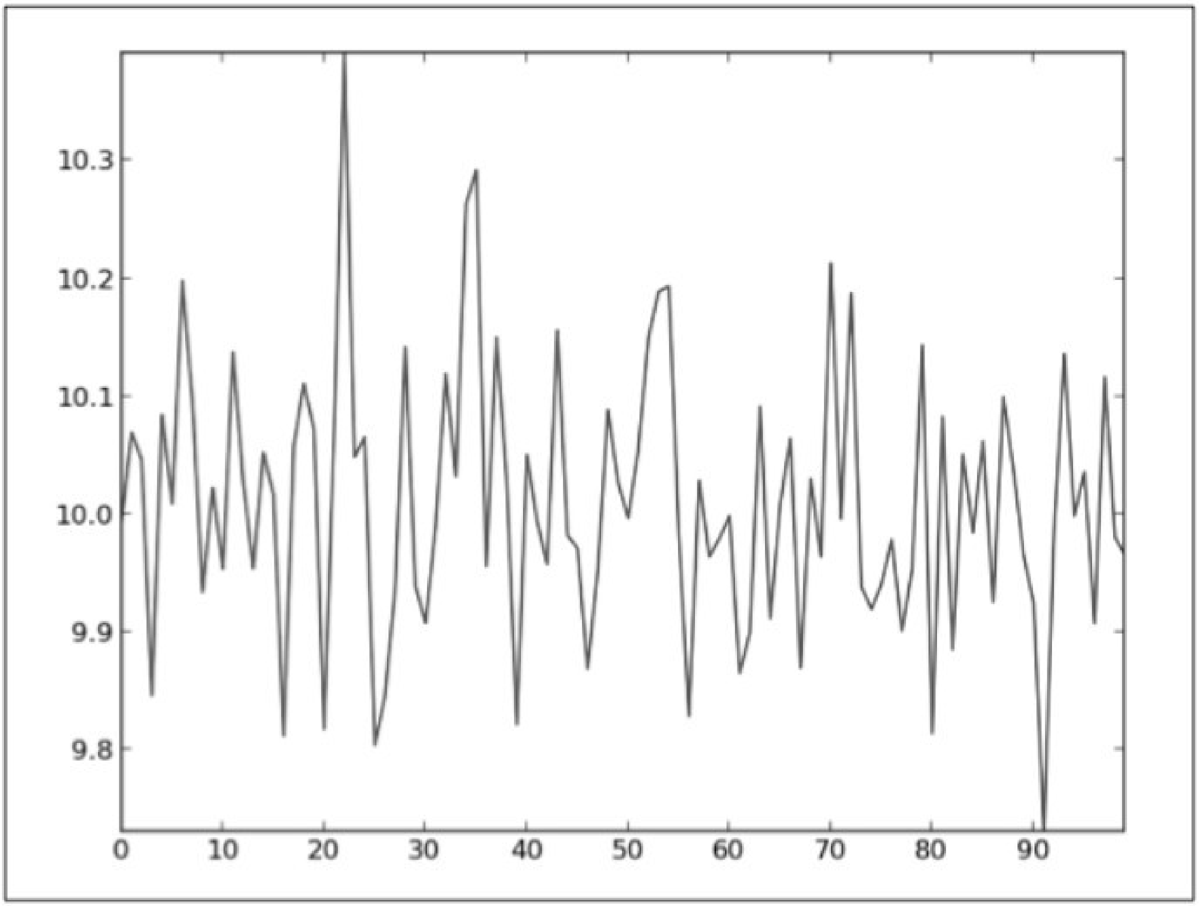
```
from pylab import *
# get current axis
ax = gca()
# set view to tight, and maximum number of tick intervals to 10
ax.locator_params(tight=True, nbins = 10)
# generate 100 normal distribution values
ax.plot(np.random.normal(10, .1, 100))
show()
```

□□□□3-8□□□□□□

□□□□□□ x □□ y □□□□□□□□□□□□□□□□□□□□□□□□□locator□□□□
□□□□□□□□□□□□□□□□□□□□10□□□□

ax.xaxis.set_major_locator(matplotlib.ticker.MultipleLocator(10))

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□matplotlib.ticker.FormatStrFormatter□□□□□□'%2.1f'□□'%1.1f cm'□□□□□□□□□□□□□



□3-8

□□□□□□□□dates□□□□□□

matplotlib □□□□□□□□□□□□□ 0001-01-01 UTC □□□□□ 1□□□□ 0001-01-01 UTC 06:00 □□□ 1.25 □

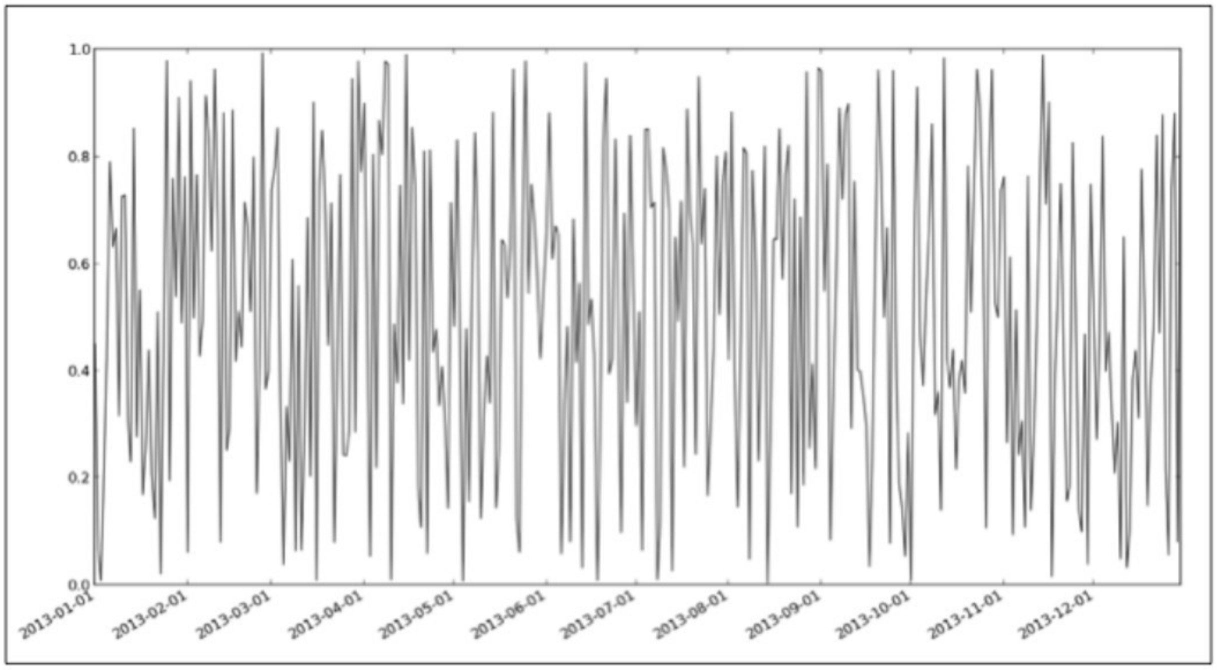
□ □ □ □ □ □ □ □ matplotlib.dates.date2num() □ matplotlib.dates.num2 date() □ matplotlib.dates.drange() □□□ helper□□□□□□□□□□□□□□□

□□□□□□□

```
from pylab import *
import matplotlib as mpl
import datetime
fig = figure()
# get current axis
ax = gca()
# set some daterange
start = datetime.datetime(2013, 01, 01)
stop = datetime.datetime(2013, 12, 31)
delta = datetime.timedelta(days = 1)
# convert dates for matplotlib
dates = mpl.dates.drange(start, stop, delta)
# generate some random values
values = np.random.rand(len(dates))
ax = gca()
# create plot with dates
ax.plot_date(dates, values, linestyle='-', marker='')
# specify formater
date_format = mpl.dates.DateFormatter('%Y-%m-%d')
# apply formater
ax.xaxis.set_major_formatter(date_format)
```

```
# autoformat date labels
# rotates labels by 30 degrees by default
# use rotate param to specify different rotation degree
# use bottom param to give more room to date labels
fig.autofmt_xdate()
show()
```

□□□□□□□□□3-9□□□□□□



□3-9

3.7 □□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□plot□□□□□□□□□□□□□□□□□□
□□

3.7.1 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

3.7.2 □□□□

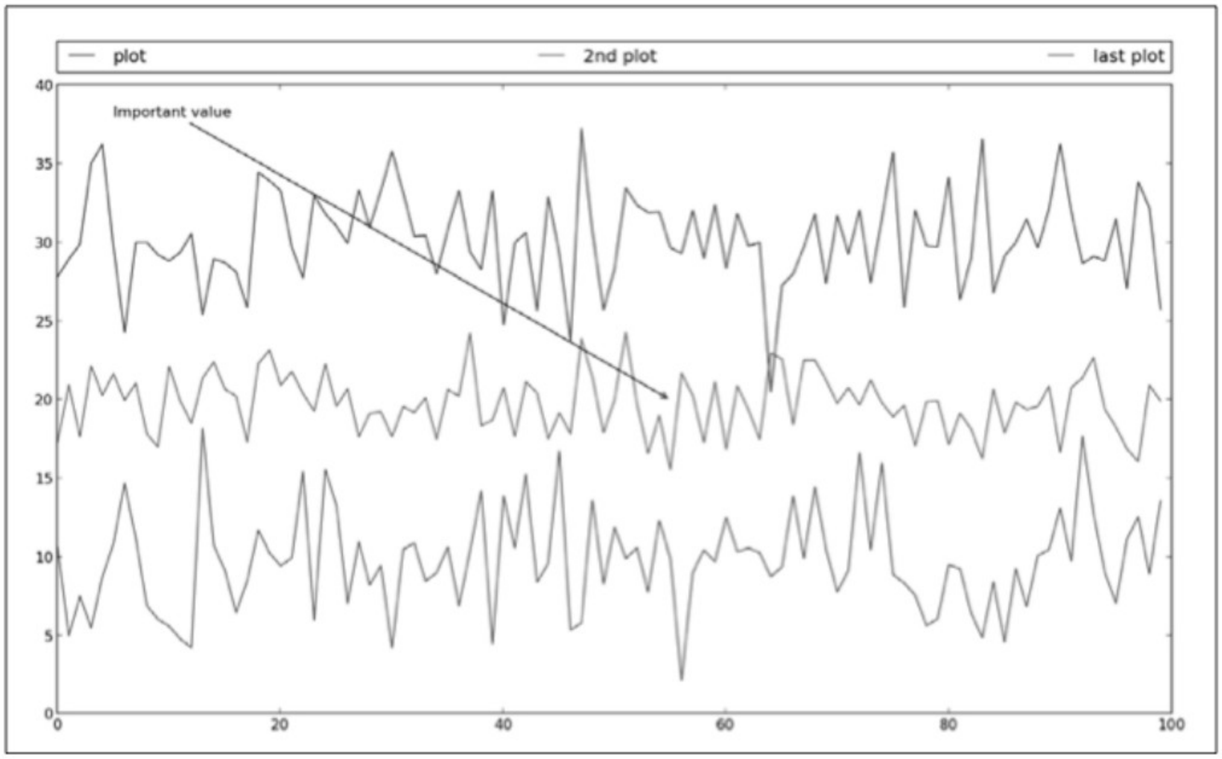
□□□□□□□□□□□□□□□□□□

```
from matplotlib.pyplot import *
# generate different normal distributions
x1 = np.random.normal(30, 3, 100)
x2 = np.random.normal(20, 2, 100)
x3 = np.random.normal(10, 3, 100)
# plot them
plot(x1, label='plot')
plot(x2, label='2nd plot')
plot(x3, label='last plot')
# generate a legend box
legend(bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
   ncol=3, mode="expand", borderaxespad=0.)
# annotate an important value
annotate("Important value", (55,20), xycoords='data',
   xytext=(5, 38),
   arrowprops=dict(arrowstyle='->'))
show()
```

□□□□□□3-10□□□□

□□□□□□□plot□□□□□□□□□□legend()□□□□□□□□□

□□□□□loc□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□



□3-10

3.7.3 □□□□

□3-3□□□□□□□□□

□3-3

字 符 串	数　值	字 符 串	数　值
upper right	1	center left	6
upper left	2	center right	7
lower left	3	lower center	8
lower right	4	upper center	9
right	5	center	10

□□□□□□□□□□□□□□□□□□□_nolegend_□

□□□□□□□□□□□□□□□□□ ncol=3□□□□□□ lower left□□□□□□□bbox_to_anchor□□□□□□(0.0, 1.02)□□□□□□□□1□□□□0.102□□□

□□□□□□□□□□□□□□mode□□□□□None□□expand□□□expand□□□□□
□□□□□□□□□□□□□□□□borderaxespad□□□□□□□□□□□□□□□□□

□□□□□□□□plot□□xy [6] □□□□□□□□□□□□□□□□□□□□□□□xycoord = 'data'□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□xytext□□□

□□□xytext□□□□xy□□□arrowprops□□□□□□□□□□□□□□□□□□□□□□□□arrowstyle□□□□□□□□□

3.8 □□□□□□□□

□□□□□□□□□□□□□□□□

□□

3.8.1 □□□□

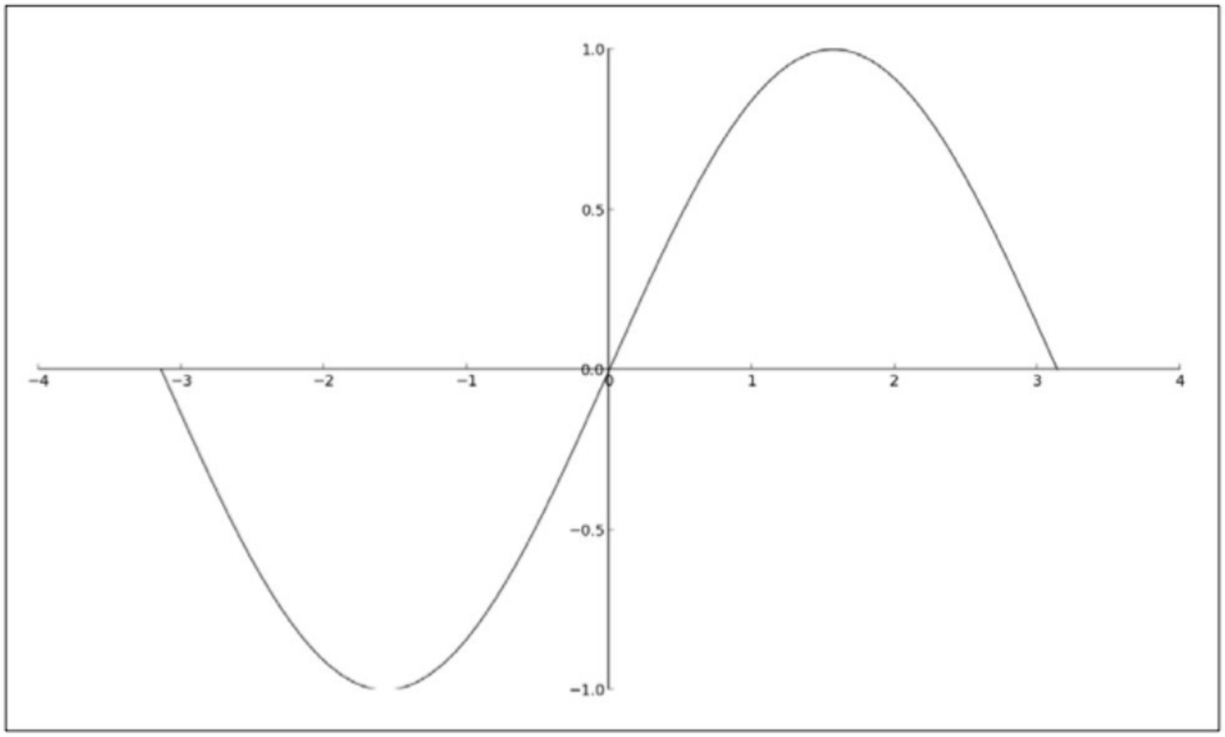
□□□□□□□□□□□□□□□□□□□□□□□□□□□□color□none□□□□□□□□□□□□□□□0□0□□□□□□□□□□□□

□□□□□□□□□□

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-np.pi, np.pi, 500, endpoint=True)
y = np.sin(x)
plt.plot(x, y)
ax = plt.gca()
# hide two spines
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
```

```
# move bottom and left spine to 0,0
ax.spines['bottom'].set_position(('data',0))
ax.spines['left'].set_position(('data',0))
# move ticks positions
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
plt.show()
```

□□□□3-11□□□□□□



□3-11

3.8.2 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□0□0□□□□□□□□□□□□□□□□0□0□□□□□□□□

□□□

3.8.3 [illegible]

[illegible] set_smart_bounds (True)[illegible]matplotlib [illegible]

3.9 [illegible]

[illegible]2D[illegible]

[illegible]bin[illegible]bin[illegible]

[illegible]1[illegible]

[illegible]

[illegible]5[illegible]“3D[illegible]”[illegible]3D[illegible]

3.9.1 [illegible]

[illegible]bin[illegible]bin[illegible]bin[illegible]ceiling[illegible]bins(k)[illegible] ceiling (max(x) – min(x)/h)[illegible] x[illegible]h[illegible]bin[illegible]bin[illegible]

3.9.2 [illegible]

[illegible] matplotlib.pyploy.hist()[illegible]

◆ bins[illegible]bin[illegible]bin[illegible]10[illegible]

◆ range□bin □□□□□ bins □□□□□□□□□□□□□□□□□□□□□□□□□□□□□None□

◆ normed□□□□□ True□□□□□□□□□□□□□□normalized□□□□□□□□□□□□□□□False□

◆ histtype□□□□ bar□□□□□□□□□□□□□□□□□

● barstacked□□□□□□□□□□□□□□

● step□□□□□□□□□□□

● stepfilled□□□□□□□□□□□□histtype□□□□ bar□

◆ align□□□bin□□□□□□□□□□□□□□□□□mid□□□□□left□right□

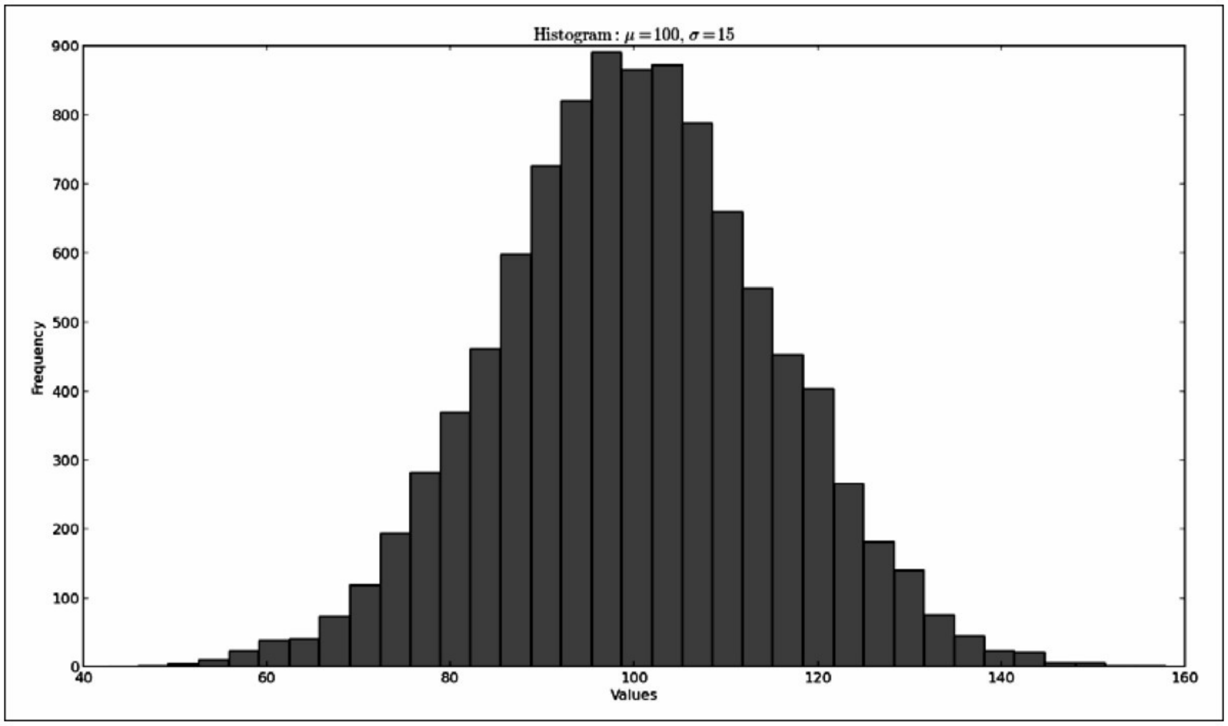
◆ color□□

◆ orientation□□□□□ orientation□ horizontal□□□□□□□□□□□□vertical□

□□□□□□□□hist()□□□□

```
import numpy as np
import matplotlib.pyplot as plt
mu = 100
sigma = 15
x = np.random.normal(mu, sigma, 10000)
ax = plt.gca()
# the histogram of the data
ax.hist(x, bins=35, color='r')
ax.set_xlabel('Values')
ax.set_ylabel('Frequency')
ax.set_title(r'$\mathrm{Histogram:}\        \mu=%d,\
\sigma=%d$' % (mu, sigma))
plt.show()
```

□□□□□□□□□□□□□□□□□□3-12□□□



□3-12

3.9.3 □□□□

□□□□□□□□□□□□□□□bin□□□35□□□□normed□True□□ 1□□□□□□□□□color□red(r)□

□□□□□□□□□□□□□□□□ matplotlib □ LaTeX □□□□□□ Python□□□□□□□□□□

3.10 □□□□□□□

□□□□□□□□□□□□□□□□

3.10.1 □□□□

[illegible]uncertainty of measurement[illegible]standard deviation[illegible]standard error[illegible]95%[illegible]confidence interval[illegible]experimental sciences[illegible]

3.10.2 [illegible]

[illegible]——left[illegible]height[illegible]

◆ width[illegible] 0.8[illegible]

◆ bottom[illegible] bottom[illegible] None[illegible]

◆ edgecolor[illegible]

◆ ecolor[illegible]

◆ linewidth[illegible] None[illegible] 0[illegible]

◆ orientation[illegible] vertical[illegible] horizontal[illegible]

◆ xerr[illegible] yerr[illegible]

[illegible]color[illegible]edgecolor[illegible]linewidth[illegible]xerr[illegible]yerr[illegible]

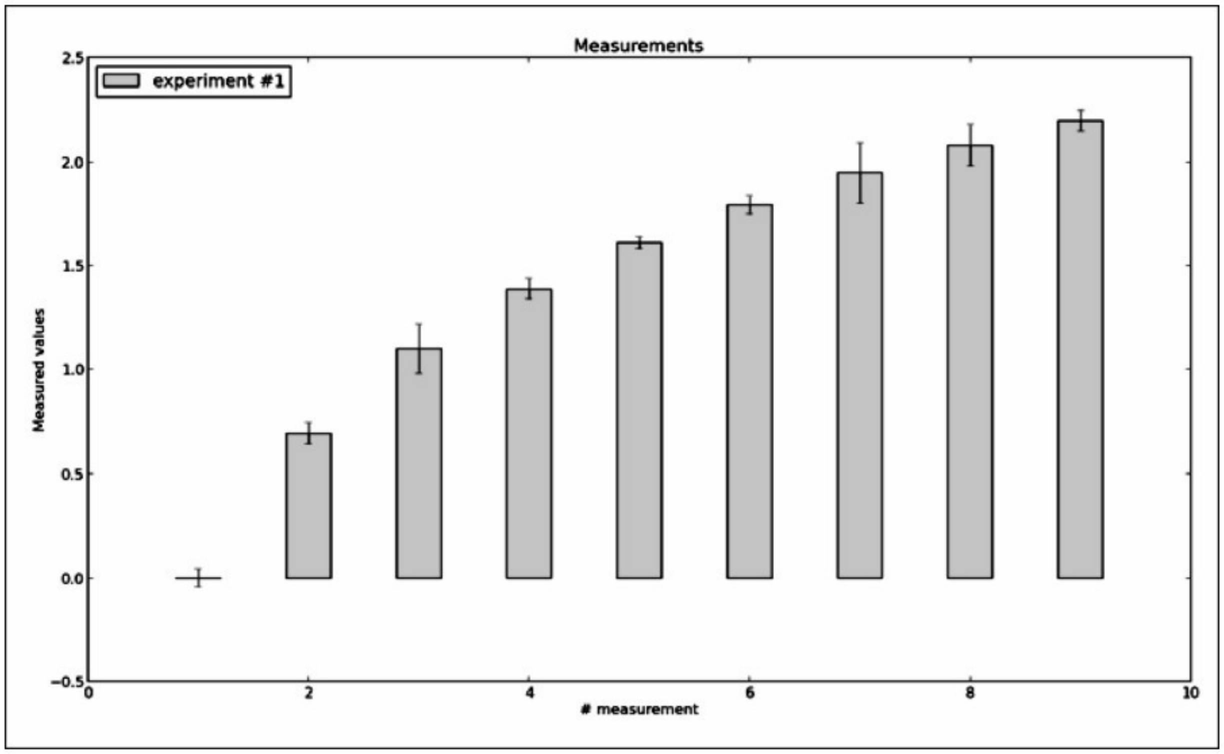
3.10.3 [illegible]

[illegible]

```
import numpy as np
import matplotlib.pyplot as plt
# generate number of measurements
x = np.arange(0, 10, 1)
# values computed from "measured"
```

```
y = np.log(x)
# add some error samples from standard normal distribution
xe = 0.1 * np.abs(np.random.randn(len(y)))
# draw and show errorbar
plt.bar(x, y, yerr=xe, width=0.4, align='center', ecolor='r',
color='cyan', label='experiment #1');
# give some explainations
plt.xlabel('# measurement')
plt.ylabel('Measured values')
plt.title('Measurements')
plt.legend(loc='upper left')
plt.show()
```

□□□□□□□3-13□□□□□



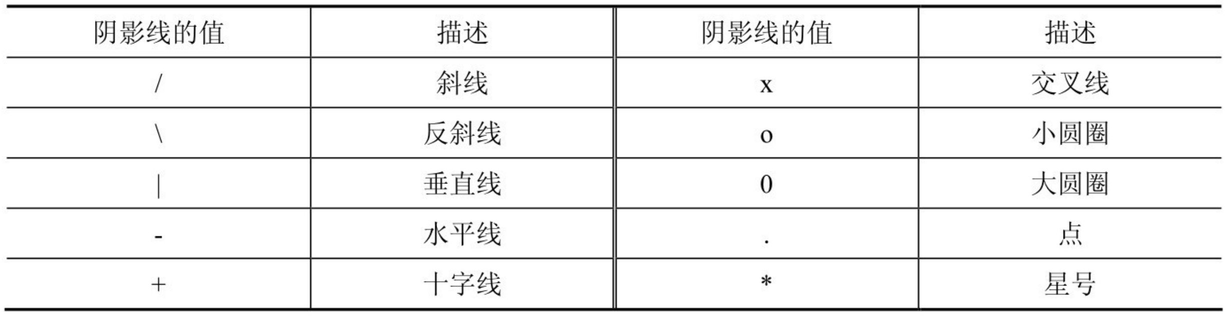
□3-13

□□□□□□□□□□□□□□□□□(x)□□□□□□□□□□□□□□□(y)□□□□□□□□(xe)□

□□□□□□NumPy□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□hatch□□□□□□□□□3-4□□□

□3-4



阴影线的值	描述	阴影线的值	描述
/	斜线	x	交叉线
\	反斜线	o	小圆圈
|	垂直线	0	大圆圈
-	水平线	.	点
+	十字线	*	星号

3.10.4 □□□□

□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□xerr□yerr□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

3.11 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□100%□□□□□□□□□□□□□□□□

3.11.1 □□□□

[illegible]

[illegible]

[illegible]

3.11.2 [illegible]

[illegible]exploded pie chart[illegible]

```
from pylab import *
# make a square figure and axes
figure(1, figsize=(6,6))
ax = axes([0.1, 0.1, 0.8, 0.8])
# the slices will be ordered
# and plotted counter-clockwise.
labels = 'Spring', 'Summer', 'Autumn', 'Winter'
# fractions are either x/sum(x) or x if sum(x) <= 1
x = [15, 30, 45, 10]
# explode must be len(x) sequence or None
explode=(0.1, 0.1, 0.1, 0.1)
pie(x, explode=explode, labels=labels,
autopct='%1.1f%%', startangle=67)
title('Rainy days by season')
show()
```

[illegible]

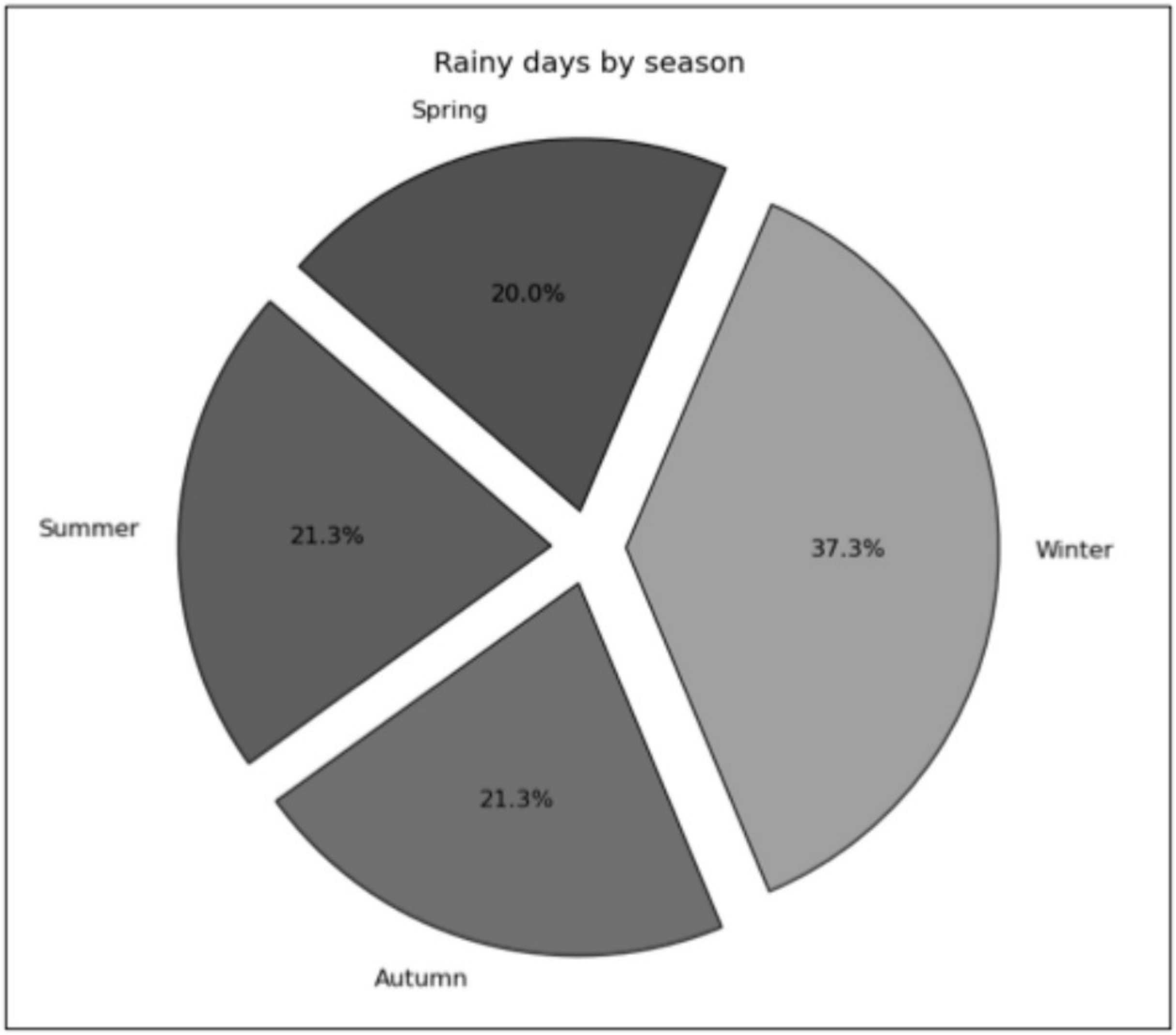
[illegible] x/sum(x)[illegible] x if sum(x) <= 1[illegible]autopct[illegible]

□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

□□□□□ startangle □□□□ x □□□ 0 □□□□□□□□□□□ atartangle□□90□□□□y□□□

□□□□□□3-14□□



□3-14

3.12 □□□□□□□□□□

[illegible]

3.12.1 [illegible]

matplotlib[illegible]

3.12.2 [illegible]

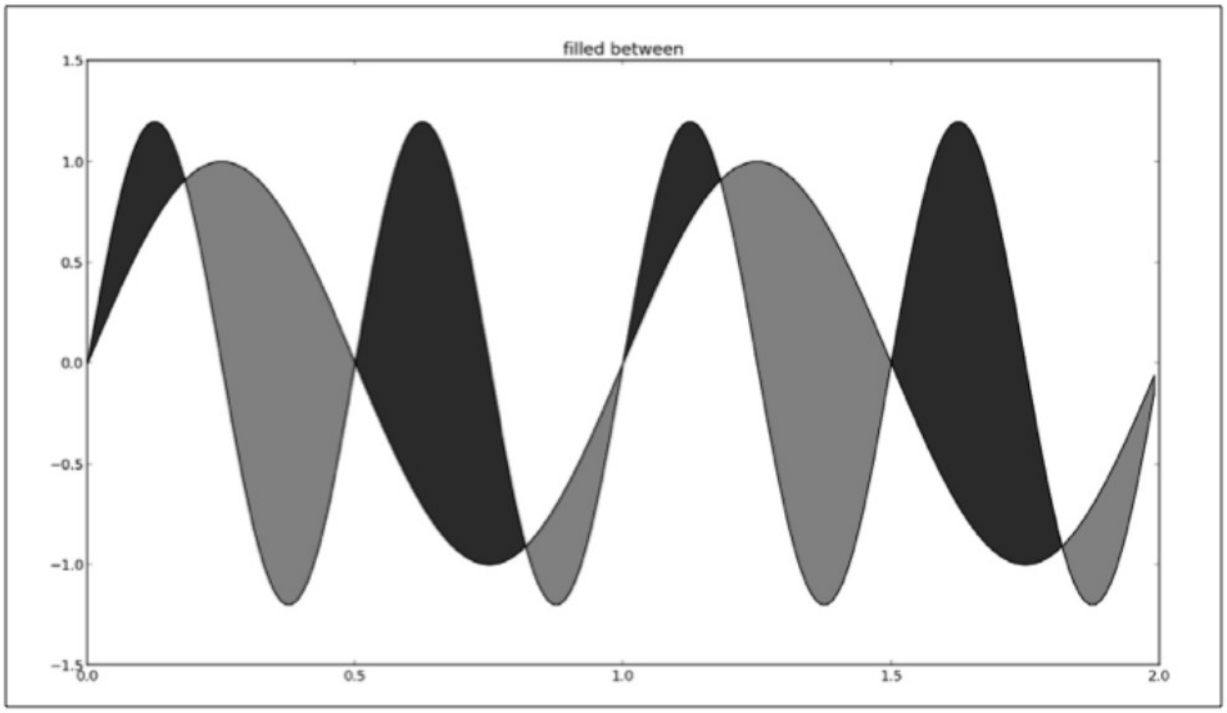
[illegible]

```
from matplotlib.pyplot import figure, show, gca
import numpy as np
x = np.arange(0.0, 2, 0.01)
# two different signals are measured
y1 = np.sin(2*np.pi*x)
y2 = 1.2*np.sin(4*np.pi*x)
fig = figure()
ax = gca()
# plot and
# fill between y1 and y2 where a logical condition is met
ax.plot(x, y1, x, y2, color='black')
ax.fill_between(x, y1, y2, where=y2>=y1, facecolor='darkblue', interpolate= True)
ax.fill_between(x, y1, y2, where=y2<=y1, facecolor='deeppink', interpolate= True)
ax.set_title('filled between')
show()
```

3.12.3 [illegible]

[illegible]plot()[illegible]fill_between()[illegible]

[illegible] 3-15 [illegible]fill_between()[illegible] x [illegible] y [illegible]y1, y2[illegible]



[illegible]3-15

[illegible]where[illegible]where[illegible]where[illegible]

3.12.4 [illegible]

[illegible]fill_between()[illegible]hatch [illegible]linewidth[illegible]linestyle[illegible]

[illegible] fill_betweenx()[illegible]

[illegible]fill()[illegible]

3.13 [illegible]

[illegible]correlation[illegible]

[illegible]scatter plot matrix[illegible]

3.13.1 [illegible]

[illegible]independent variable[illegible]dependent variable[illegible] y [illegible]

3.13.2 [illegible]

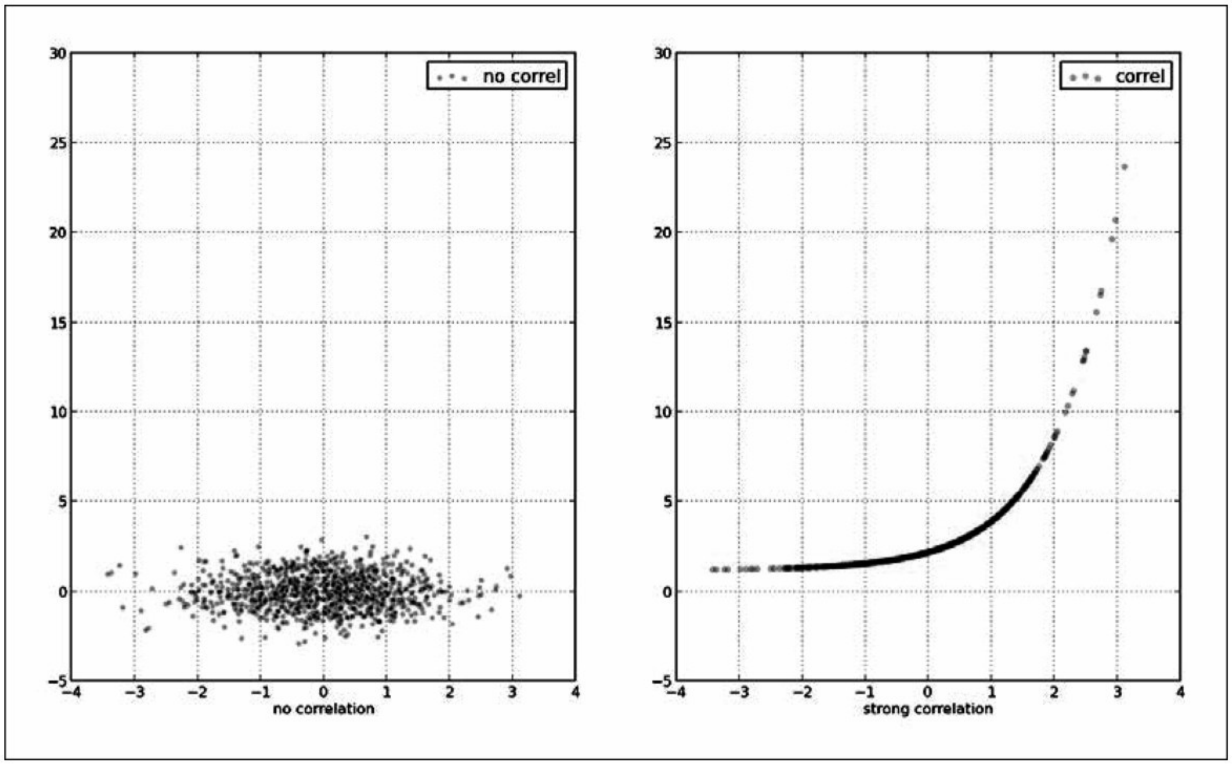
[illegible]strong positive correlation[illegible]

```
import matplotlib.pyplot as plt
import numpy as np
# generate x values
x = np.random.randn(1000)
# random measurements, no correlation
y1 = np.random.randn(len(x))
# strong correlation
y2 = 1.2 + np.exp(x)
ax1 = plt.subplot(121)
plt.scatter(x, y1, color='indigo', alpha=0.3, edgecolors='white',
label='no correl')
```

```
    plt.xlabel('no correlation')
    plt.grid(True)
    plt.legend()
    ax2 = plt.subplot(122, sharey=ax1, sharex=ax1)
    plt.scatter(x, y2, color='green', alpha=0.3, edgecolors='grey',
    label='correl')
    plt.xlabel('strong correlation')
    plt.grid(True)
    plt.legend()
    plt.show()
```

□□□□□□□□□□□□□□□□□□□□□□□□□color□□□□□□□□□□□□□circle□□marker□alpha□alpha□□□□□edgecolors□□□□□□□□□□label□□□□□□□□

□□□□3-16□□□□□□



[illegible]3-16

3.13.3 [illegible]

[illegible]matplotlib[illegible]scatter()[illegible] x [illegible]unidimensional array[illegible]y [illegible]

[illegible]

[1].[illegible] Mac OS X [illegible] ishold()[illegible] True[illegible]

[2].[illegible] Pi[illegible]

[3]. [illegible] matplotlib.pyploy.autoscale()[illegible]

[4].[illegible] axspan[illegible]

[5]. [illegible] subplots [illegible]Python [illegible]figure [illegible] axes [illegible]

[6]. annotate [illegible]

[illegible]4[illegible] [illegible]

[illegible]

- ◆ [illegible]
- ◆ [illegible]
- ◆ [illegible]
- ◆ [illegible] subplots[illegible]
- ◆ [illegible]
- ◆ [illegible]
- ◆ [illegible]
- ◆ [illegible]
- ◆ [illegible]

4.1 [illegible]

[illegible]matplotlib[illegible]

[illegible]

4.2 [illegible]

Axes[illegible]axes[illegible]
[illegible]

4.2.1 [illegible]

[illegible]matplotlib[illegible]

[illegible] Figure[illegible] Figure[illegible]Axes[illegible]Figure.axes[illegible]Axes[illegible]plot()[illegible]Axes.lines[illegible]matplotlib.lines.Line2D[illegible]hist()[illegible]Axes.patches[illegible]“patches[1]”[illegible]MATLAB™ [illegible]“[illegible]”[illegible]

Axes[illegible]XAxis[illegible]YAxis[illegible]x[illegible]y[illegible]XAxis[illegible]YAxis[illegible]Axes.xaxis[illegible]Axes.yaxis[illegible]XAxis[illegible]YAxis[illegible]matplotlib[illegible]helper[illegible]matplotlib.pyplot.xlabel()[illegible]matplotlib. pyplot.ylabel()[illegible]

4.2.2 [illegible]

[illegible]

1.[illegible]plot[illegible]

2.[illegible]title[illegible]axes[illegible]

3.[illegible]alpha[illegible]

4.[illegible]title[illegible]axes[illegible]

```
import matplotlib.pyplot as plt
from matplotlib import patheffects
import numpy as np
```

```
data = np.random.randn(70)
fontsize = 18
plt.plot(data)
title = "This is figure title"
x_label = "This is x axis label"
y_label = "This is y axis label"
title_text_obj = plt.title(title, fontsize=fontsize,
verticalalignment='bottom')
title_text_obj.set_path_effects([patheffects.
withSimplePatchShadow()])
# offset_xy -- set the 'angle' of the shadow
# shadow_rgbFace -- set the color of the shadow
# patch_alpha -- setup the transparency of the shadow
offset_xy = (1, -1)
rgbRed = (1.0,0.0,0.0)
alpha = 0.8
# customize shadow properties
pe = patheffects.withSimplePatchShadow(offset_xy =
offset_xy,
    shadow_rgbFace = rgbRed,
    patch_alpha = alpha)
# apply them to the xaxis and yaxis labels
xlabel_obj = plt.xlabel(x_label, fontsize=fontsize,
alpha=0.5)
xlabel_obj.set_path_effects([pe])
ylabel_obj = plt.ylabel(y_label, fontsize=fontsize,
alpha=0.5)
```

ylabel_obj.set_path_effects([pe])
plt.show()

4.2.3 □□□□

□□□□□□□□□□□□imports□□2□□“□□□□□□”□□□□3□□“□□□□□□□□”□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□bottom□□□□□□□□□□matplotlib.patheffects.withSimple PatchShadow()□□□□□□□□□□□□□□□□□□□□□□offset_xy=(2,-2)□shadow_rgbFace=None□patch_alpha=0.7□□□□□□□□□□□□□□center□top□baseline□□□□□□□□□□□□□□□□□□□□bottom□□□□□□□□□□□□□□□□□□□□□□□□□□path effects□□ matplotlib □ matplotlib. patheffects□□□□□□□□□□□□matplotlib.text.Text□ matplotlib. patches.Patch□

□□□x□□y□□□□□□□□□□□□□□□□□□□□□□□□□[2]□□□□□□□offset□□□□□□□□□□□□□□□□□□□□□□□□□ 0.0□1.0 □□□□□□□□□□□□□□□□□□□□RGB□□□□□□□□□□□□1.0□0.0□0.0□□□□□□□□□□□□□□

□□□□□□alpha□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□ □ □ □ □ □ □ □ □ matplotlib.patheffects.withSimplePatchShadow□□,□□□□□□□□□pe□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□label□□□□□□□□□□□□□□□matplotlib. pyplot.xlabel()□□□□□□□□matplotlib.text.Text□□□□□□□□□□□□□□set_path_effects([pe])□□□

□□□□□□□□□□□□□□□□□□□□□□□

4.2.4 □□□□

[illegible] matplotlib.patheffects [illegible] matplotlib.pathefffects._Base[illegible]draw_path[illegible]

https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/patheffe cts.py#L47

4.3 [illegible]

[illegible]

4.3.1 [illegible]

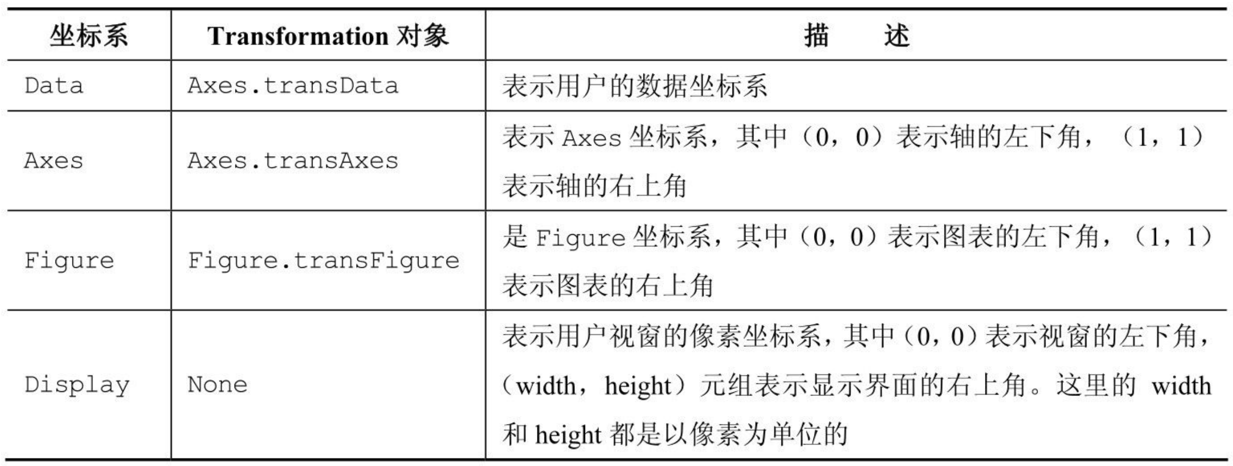
[illegible]matplotlib[illegible]transformation[illegible]matplotlib.transforms[illegible]

[illegible]matplotlib[illegible]transformations[illegible]

Transformations [illegible]

[illegible]4-1[illegible]

[illegible]4-1



坐标系	Transformation 对象	描　　述
Data	Axes.transData	表示用户的数据坐标系
Axes	Axes.transAxes	表示 Axes 坐标系，其中（0，0）表示轴的左下角，（1，1）表示轴的右上角
Figure	Figure.transFigure	是 Figure 坐标系，其中（0，0）表示图表的左下角，（1，1）表示图表的右上角
Display	None	表示用户视窗的像素坐标系，其中（0，0）表示视窗的左下角，（width，height）元组表示显示界面的右上角。这里的 width 和 height 都是以像素为单位的

[illegible]Transformations[illegible]Display[illegible] Figure[illegible]Axes [illegible] Data [illegible]

[illegible]

4.3.2 [illegible]

[illegible]

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.transforms as transforms
def setup(layout=None):
   assert layout is not None
   fig = plt.figure()
   ax = fig.add_subplot(layout)
   return fig, ax
def get_signal():
   t = np.arange(0., 2.5, 0.01)
   s = np.sin(5 * np.pi * t)
   return t, s
```

```
    def plot_signal(t, s):
        line, = axes.plot(t, s, linewidth=5, color='magenta')
        return line,
    def make_shadow(fig, axes, line, t, s):
        delta = 2 / 72. # how many points to move the shadow
        offset = transforms.ScaledTranslation(delta, -delta,
    fig.dpi_scale_trans)
        offset_transform = axes.transData + offset
        # We plot the same data, but now using offset
  transform
        # zorder -- to render it below the line
        axes.plot(t, s, linewidth=5, color='gray',
            transform=offset_transform,
            zorder=0.5 * line.get_zorder())
    if __name__ == "__main__":
        fig, axes = setup(111)
        t, s = get_signal()
        line, = plot_signal(t, s)
        make_shadow(fig, axes, line, t, s)
        axes.set_title('Shadow effect using an offset
  transform')
        plt.show()
```

4.3.3 [illegible]

[illegible] if__name__[illegible] setup()[illegible]figure[illegible] axes[illegible]plot_signal()[illegible]

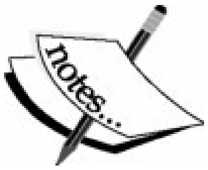
□□□□□□□□□□□□□□□make_shadow()□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□plot()□□□□□□□

matplotlib □ □ □ □ transformations helper——matplotlib.transforms.Scaled Translation——□□□□□□□□

dx□dy□□□□□□□□□□□□1/72□□□□□□□□□□□2pt□□□□□□□□□2pt□



□□□□□□□□□□□□□□□1/72 [3] □□□□□□□□□□□□□Wikipedia □□□□□http://en.wikipedia.org/wiki/ Point_%28 typography%29□

□□□□matplotlib.transforms.ScaledTransformation(xtr, ytr, scaletr)□□□□□□xtr□ytr□□□□□□□□scaletr□□□□□□□□□□□callable□□□□□□□□□□□□xtr□ytr□□□□□□□□□□□□□□□□□□□□□□□□□□ DPI□□□matplotlib□□□□□Figure.dpi_scale_trans□□□

□□□□□□□□□□□□□□□□

4.3.4 □□□□

□□ transforms □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ trans formations□□□□□□□□□□□□□transformation□□matplotlib□□□matplotlib□□□□□□□□□□□□□□

4.4 □□□□□□□□

[illegible]matplotlib[illegible]

4.4.1 [illegible]

[illegible]

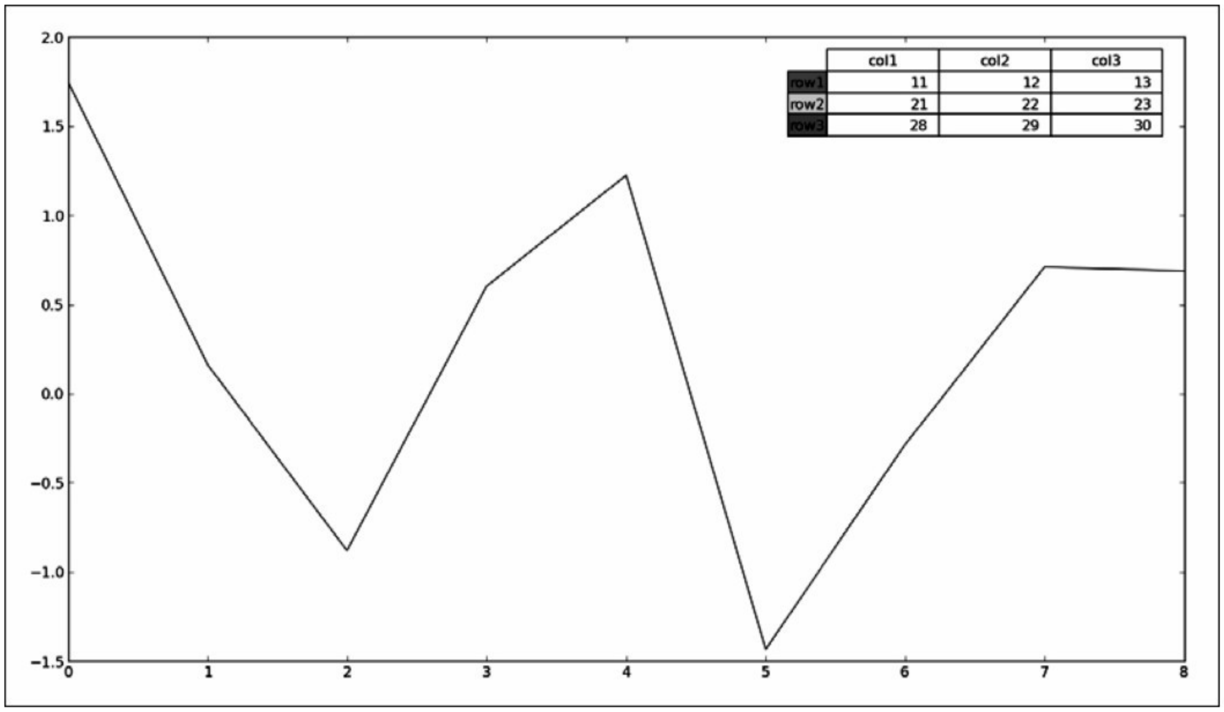
[illegible]USD[illegible]

4.4.2 [illegible]

[illegible]

```
import matplotlib.pylab as plt
import numpy as np
plt.figure()
ax = plt.gca()
y = np.random.randn(9)
col_labels = ['col1','col2','col3']
row_labels = ['row1','row2','row3']
table_vals = [[11, 12, 13], [21, 22, 23], [28, 29, 30]]
row_colors = ['red', 'gold', 'green']
my_table = plt.table(cellText=table_vals,
   colWidths=[0.1] * 3,
   rowLabels=row_labels,
   colLabels=col_labels,
   rowColours=row_colors,
    loc='upper right')
plt.plot(y)
plt.show()
```

[illegible]4-1[illegible]



	col1	col2	col3
row1	11	12	13
row2	21	22	23
row3	28	29	30

[illegible]4-1

4.4.3 [illegible]

[illegible]plt.table()[illegible]

[illegible]

```
table(cellText=None, cellColours=None,
    cellLoc='right', colWidths=None,
    rowLabels=None, rowColours=None, rowLoc='left',
    colLabels=None, colColours=None, colLoc='center',
```

loc='bottom', bbox=None)

[illegible] matplotlib.table.Table[illegible] matplotlib [illegible]add_table()[illegible] matplotlib. table.Table[illegible]

4.4.4 [illegible]

[illegible]matplotlib.table.Table[illegible]axes[illegible] Axes.add_table(table)[illegible] table [illegible]axes[illegible] table[illegible]matplotlib.table.Table[illegible]

4.5 [illegible]subplots([illegible])

[illegible]subplot[illegible]subplot[illegible]axes[illegible] subplot[illegible]

[illegible]plot[illegible]

4.5.1 [illegible]

[illegible] matplotlib.axes.SubplotBase [illegible] matplotlib. axes.Axes[illegible]helper[illegible]Axes[illegible]

[illegible]matplotlib.figure.SubplotParams[illegible]subplot[illegible]subplot[illegible]rc[illegible]

[illegible]matplotlib.pyplot[illegible]helper[illegible]

matplotlib.pyplot.subplots [illegible]——[illegible]

[illegible]x[illegible]y[illegible]sharex[illegible]sharey[illegible]sharex[illegible]True[illegible]x[illegible]row[illegible]col[illegible]all[illegible]none[illegible]all[illegible]True[illegible]none[illegible]False[illegible]row[illegible]x[illegible]col[illegible]y[illegible]matplotlib.pyplot. subplots[illegible]fig[illegible]ax[illegible]ax[illegible]ax[illegible]

[illegible]matplotlib.pyplot.subplots_adjust[illegible]left[illegible]right[illegible] bottom[illegible] top)[illegible]wspace[illegible] hspace[illegible]

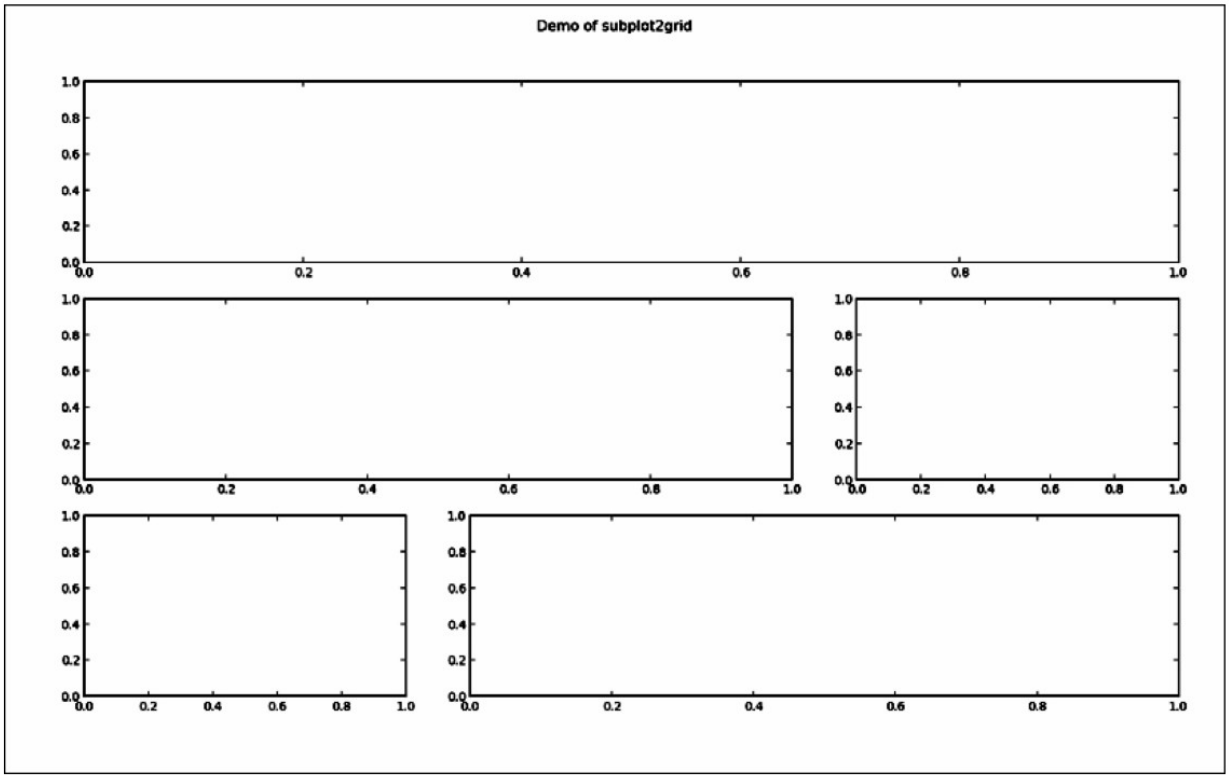
4.5.2 [illegible]

[illegible]matplotlib[illegible]helper[illegible]——subplot2grid——[illegible]0[illegible]plot.subplot()[illegible]1[illegible]colspan[illegible]rowspan[illegible]subplot2grid[illegible]

[illegible]

```
import matplotlib.pyplot as plt
plt.figure(0)
axes1 = plt.subplot2grid((3, 3), (0, 0), colspan=3)
axes2 = plt.subplot2grid((3, 3), (1, 0), colspan=2)
axes3 = plt.subplot2grid((3, 3), (1, 2))
axes4 = plt.subplot2grid((3, 3), (2, 0))
axes5 = plt.subplot2grid((3, 3), (2, 1), colspan=2)
# tidy up tick labels size
all_axes = plt.gcf().axes
for ax in all_axes:
    for ticklabel in ax.get_xticklabels() +
ax.get_yticklabels():
        ticklabel.set_fontsize(10)
  plt.suptitle("Demo of subplot2grid")
  plt.show()
```

□□□□□□□□□□□□□□□□4-2□□□□□□



□4-2

4.5.3 □□□□

□subplot2grid□□□□□□□□□□□□□loc□□□□□□□rowspan□colspan□□□□□□□□□□□□□□□□□0□□□□□□figure.add_subplot□1□□□□

4.5.4 □□□□

[illegible]axes[illegible]subplot[illegible]

```
axes = fig.add_subplot(111)
rectangle = axes.patch
rectangle.set_facecolor('blue')
```

[illegible]axes[illegible]rectangle[illegible]patch[illegible]axes[illegible]axes[illegible]

[illegible]axes[illegible]

```
fig = plt.figure()
axes = fig.add_subplot(111)
rect = matplotlib.patches.Rectangle((1,1), width=6, height=12)
axes.add_patch(rect)
# we have to manually force a figure draw
axes.figure.canvas.draw()
```

4.6 [illegible]

[illegible] matplotlib.pyplot.grid[illegible]

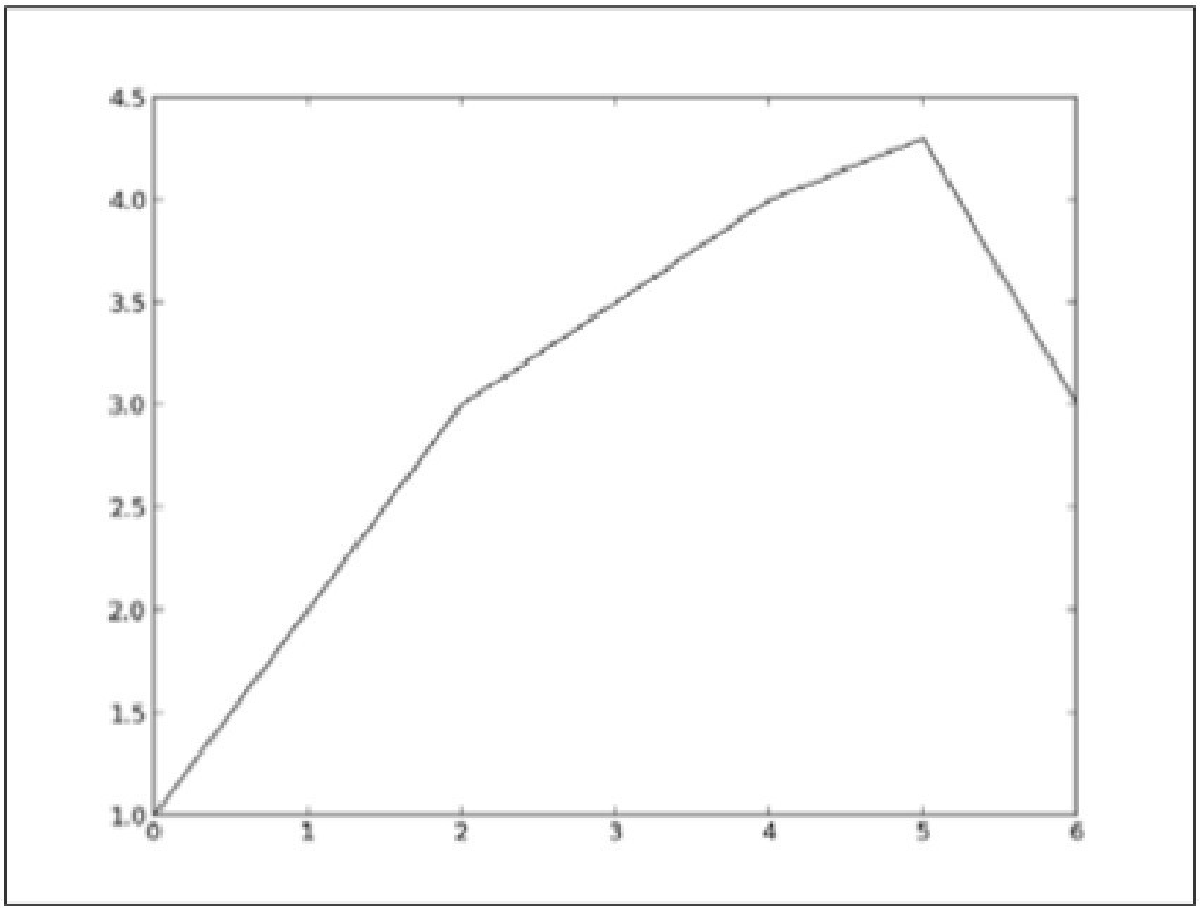
[illegible]

4.6.1 [illegible]

[illegible] matplotlib.pyplot.grid helper [illegible]

□□□□□□□□□□□□ipython–pylab□□□□□□□□□□□plt.grid()□□□□□□□□
□□ IPython PyLab □□□□□□□□□□□□□□□□□□□□□□□□□□ 4-3□□□

```
In [1]: plt.plot([1,2,3,3.5,4,4.3,3])
Out[1]: [<matplotlib.lines.Line2D at 0x3dcc810>]
```

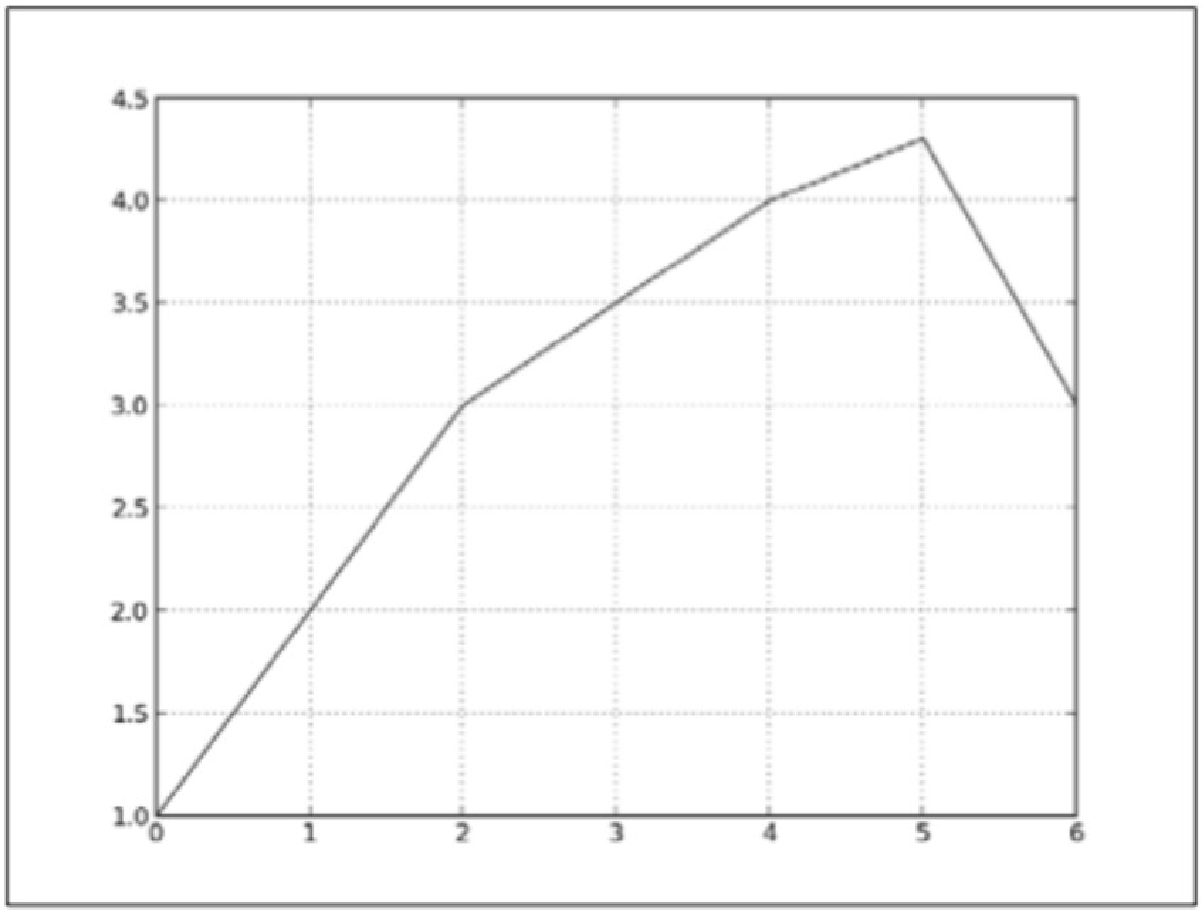


□4-3

□□□□□□□□□□□□□□□□□□□

```
In [2]: plt.grid()
```

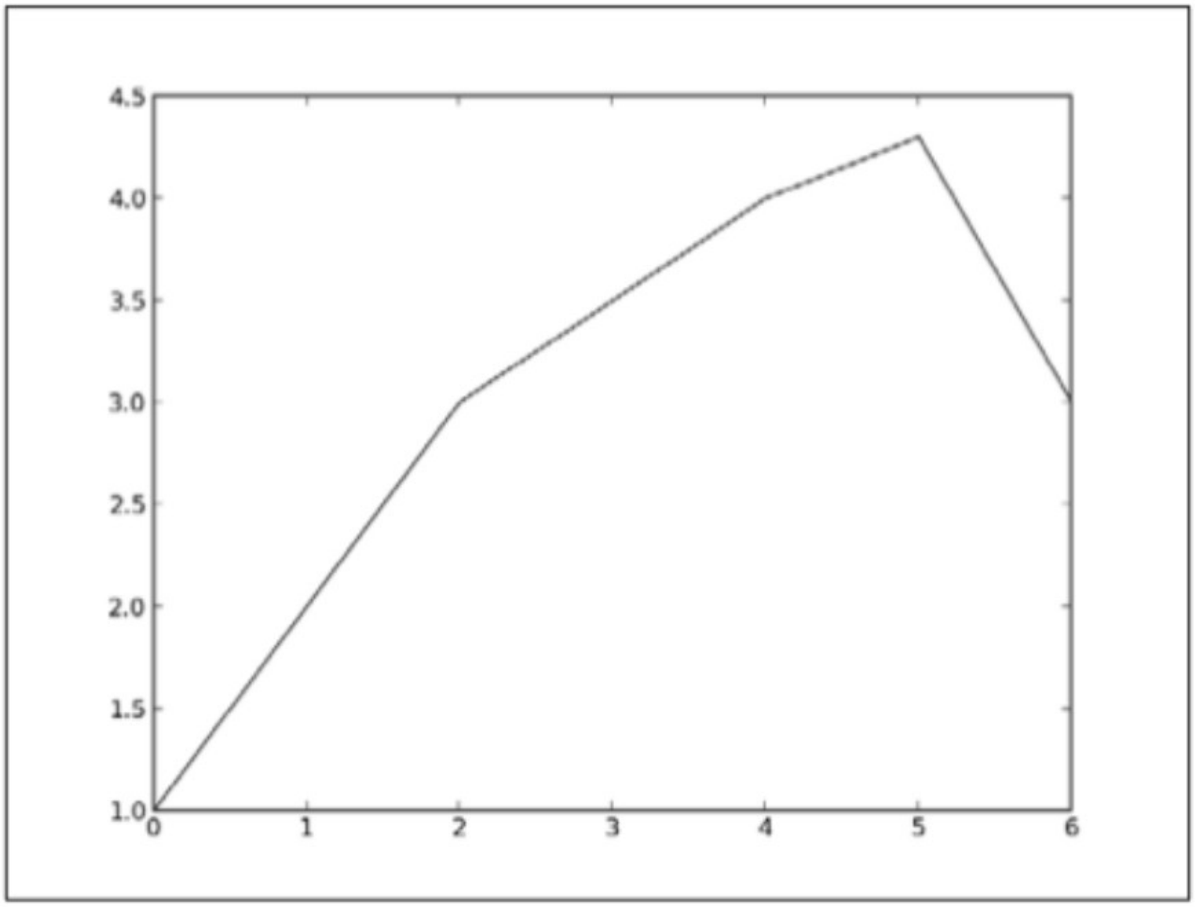
□□□□□□□□4-4□□□



[illegible]4-4

[illegible]4-5[illegible]

```
In [3]: plt.grid()
```



□4-5

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□which□□□'major'□'minor'□□□'both'□□□□□□□□□□□□axis□□□□□□□□□□□□□□□□□'x'□'y'□□□'both'□

□□□□□□□kwargs□□□□□□□□matplotlib.lines.Line2D□□□□□□□□□□□□□color□linestyle□linewidth□□□□□□□□

ax.grid(color='g', linestyle='--', linewidth=1)

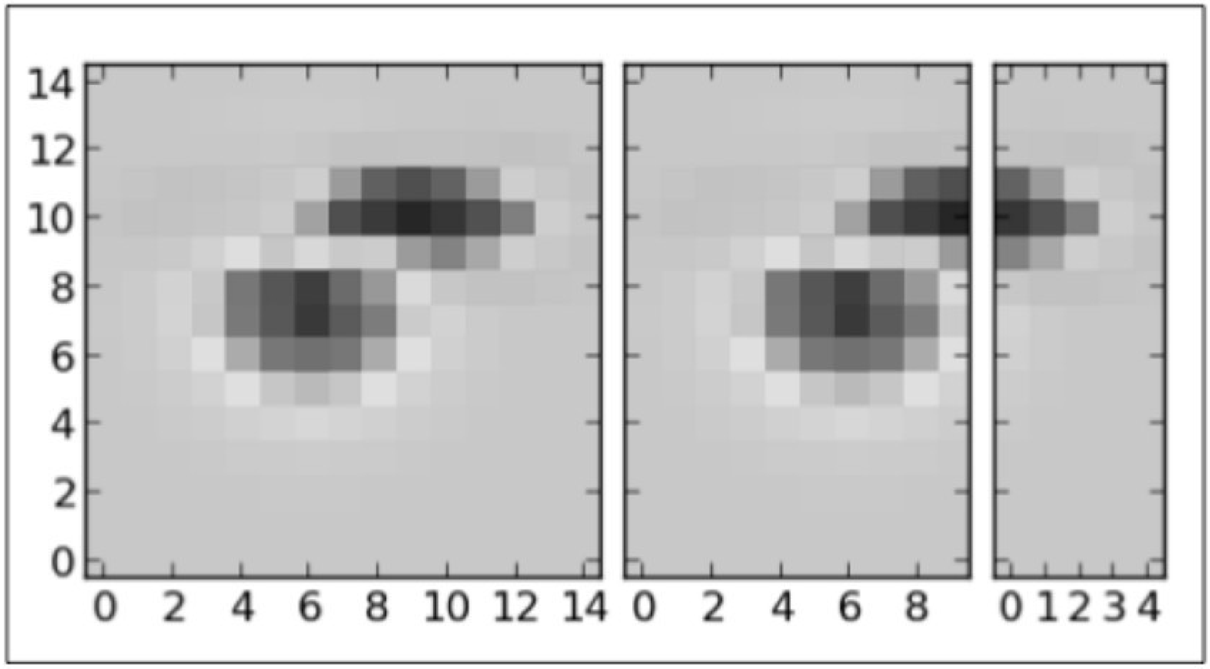
4.6.2 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□matplotlib□mpl_toolkits□□□□□□□□□□□□□□□□□□□□□AxesGrid□□□

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import ImageGrid
from matplotlib.cbook import get_sample_data
def get_demo_image():
    f = get_sample_data("axes_grid/bivariate_normal.npy",
asfileobj=False)
    # z is a numpy array of 15x15
    Z = np.load(f)
    return Z, (-3, 4, -4, 3)
def              get_grid(fig=None,              layout=None,
nrows_ncols=None):
    assert fig is not None
    assert layout is not None
    assert nrows_ncols is not None
    grid            =            ImageGrid(fig,            layout,
  nrows_ncols=nrows_ncols,
        axes_pad=0.05, add_all=True, label_mode="L")
    return grid
def load_images_to_grid(grid, Z, *images):
    min, max = Z.min(), Z.max()
    for i, image in enumerate(images):
      axes = grid[i]
      axes.imshow(image,     origin="lower",     vmin=min,
vmax=max,
        interpolation="nearest")
if __name__ == "__main__":
```

```
fig = plt.figure(1, (8, 6))
grid = get_grid(fig, 111, (1, 3))
Z, extent = get_demo_image()
# Slice image
image1 = Z
image2 = Z[:, :10]
image3 = Z[:, 10:]
load_images_to_grid(grid, Z, image1, image2, image3)
plt.draw()
plt.show()
```

□□□□□□□□□4-6□□□□□□



□4-6

4.6.3 □□□□

□□□get_demo_image□□□□□matplotlib□□□□□□□□□□□□□ grid□□□□□axes□□□□□□□ImageGrid□□

[illegible]image1[illegible]image2[illegible]image3[illegible]Z[illegible]grid[illegible]

[illegible]imshow()[illegible]image1[illegible]image2[illegible]image3 [4] [illegible]matplotlib[illegible]

4.7 [illegible]

[illegible]contour plot[illegible]isolines[illegible] [5] [illegible]

[illegible]

4.7.1 [illegible]

Z[illegible]Z[illegible]X-Y[illegible]Z[illegible]2[illegible]

[illegible]

[illegible]clabel()[illegible]colormaps[illegible]colormaps [illegible]

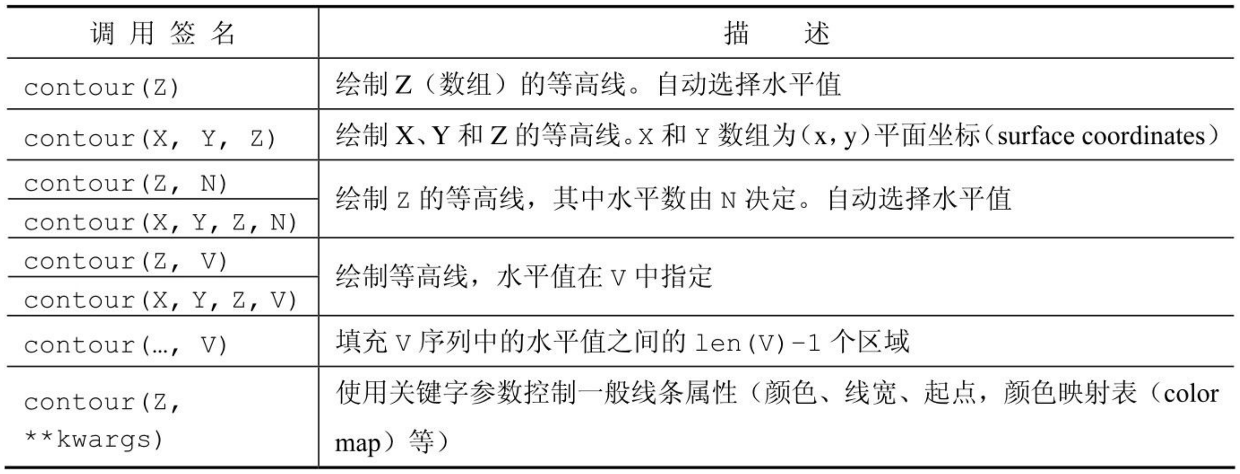
[illegible]

[illegible]contour()[illegible]

[illegible]matplotlib[illegible]matplotlib.pyplot.contour[illegible]

[illegible]contour()[illegible]contourf()[illegible]contour()[illegible]contourf()[illegible] contour()[illegible]4-2[illegible]

[illegible]4-2



调 用 签 名	描 述
contour(Z)	绘制 Z（数组）的等高线。自动选择水平值
contour(X, Y, Z)	绘制 X、Y 和 Z 的等高线。X 和 Y 数组为（x，y）平面坐标（surface coordinates）
contour(Z, N)	绘制 Z 的等高线，其中水平数由 N 决定。自动选择水平值
contour(X, Y, Z, N)	
contour(Z, V)	绘制等高线，水平值在 V 中指定
contour(X, Y, Z, V)	
contour(…, V)	填充 V 序列中的水平值之间的 len(V)-1 个区域
contour(Z, **kwargs)	使用关键字参数控制一般线条属性（颜色、线宽、起点，颜色映射表（color map）等）

X[illegible]Y[illegible]Z[illegible]X[illegible]Y[illegible]Z[illegible]X[illegible]Z[illegible]Y[illegible]Z[illegible]

4.7.2 [illegible]

[illegible]

1.[illegible]

2.[illegible]

3.[illegible]

4.[illegible]

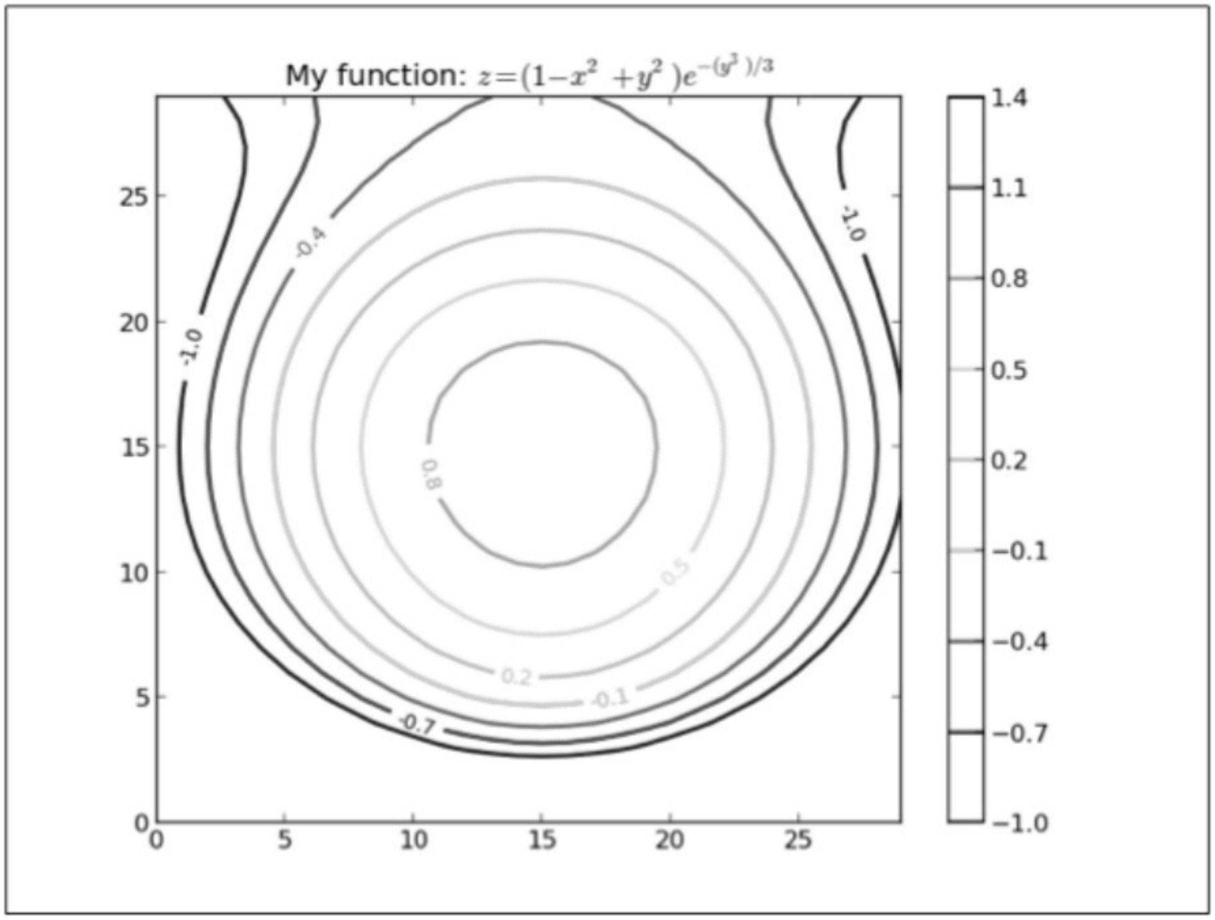
5.[illegible]

6.[illegible]

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
def process_signals(x, y):
    return (1 - (x ** 2 + y ** 2)) * np.exp(-y ** 3 / 3)
```

```
x = np.arange(-1.5, 1.5, 0.1)
y = np.arange(-1.5, 1.5, 0.1)
# Make grids of points
X, Y = np.meshgrid(x, y)
Z = process_signals(X, Y)
# Number of isolines
N = np.arange(-1, 1.5, 0.3)
# adding the Contour lines with labels
CS = plt.contour(Z, N, linewidths=2, cmap=mpl.cm.jet)
plt.clabel(CS, inline=True, fmt='%1.1f', fontsize=10)
plt.colorbar(CS)
plt.title('My function: $z=(1-x^2+y^2) e^{-(y^3)/3}$')
plt.show()
```

□□□□4-7□□□□□□



□4-7

4.7.3 □□□□

□□□ numpy□□□□□□ helper □□□□□□□□□□□

□□my_function [6] □□□□□□Z□□□□□□□□contour□□□□Z□□□□□□□□□

□□□□□□□□ N arange()□□□□□□□□□□□□□□□□□□□□N=np.arange(-1, 1.5, 0.3)□□□□□□□□□□□ 0.3□□ 0.1□ 1□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□CS□matplotlib.contour.QuadContourSet□□□□□□□□□□□□□□□□□

4.8 [illegible]

[illegible]matplotlib[illegible]matplotlib.pyplot.fill[illegible]

[illegible]matplotlib.pyplot.plot[illegible]x[illegible]y[illegible]Line2D[illegible]Patch[illegible]

[illegible]

4.8.1 [illegible]

[illegible] histogram()[illegible]matplotlib [illegible]

[illegible]——matplotlib.pyplot.fill [illegible] matplotlib.pyplot.fill_between()[illegible]matplotlib.pyplot.fill_betweenx()[7][illegible]fill_between()[illegible]fill_betweenx()[illegible]x[illegible]y[illegible]

[illegible]fill_between[illegible]x[illegible]x[illegible]y1[illegible]y2[illegible]y[illegible]y[illegible]None[illegible]

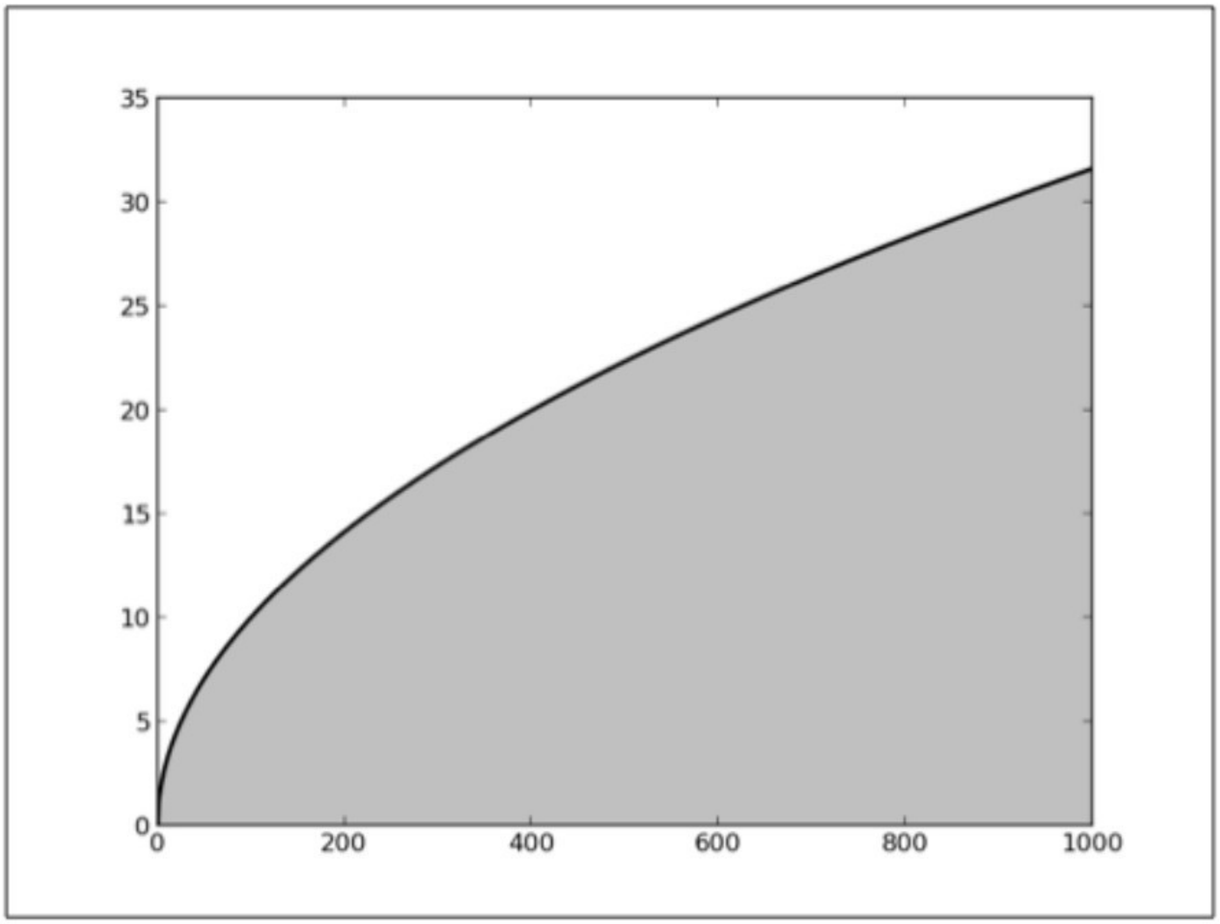
4.8.2 [illegible]

[illegible]

```
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt
t = range(1000)
y = [sqrt(i) for i in t]
plt.plot(t, y, color='red', lw=2)
plt.fill_between(t, y, color='silver')
plt.show()
```

□□□□□□□□4-8□□□□□□

□□□□□□□□□□□□ fill_between() □□□□□□□□□□□□□□ fill_between()□□□□□□□□□□□□□□'silver'□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□plot□□□□



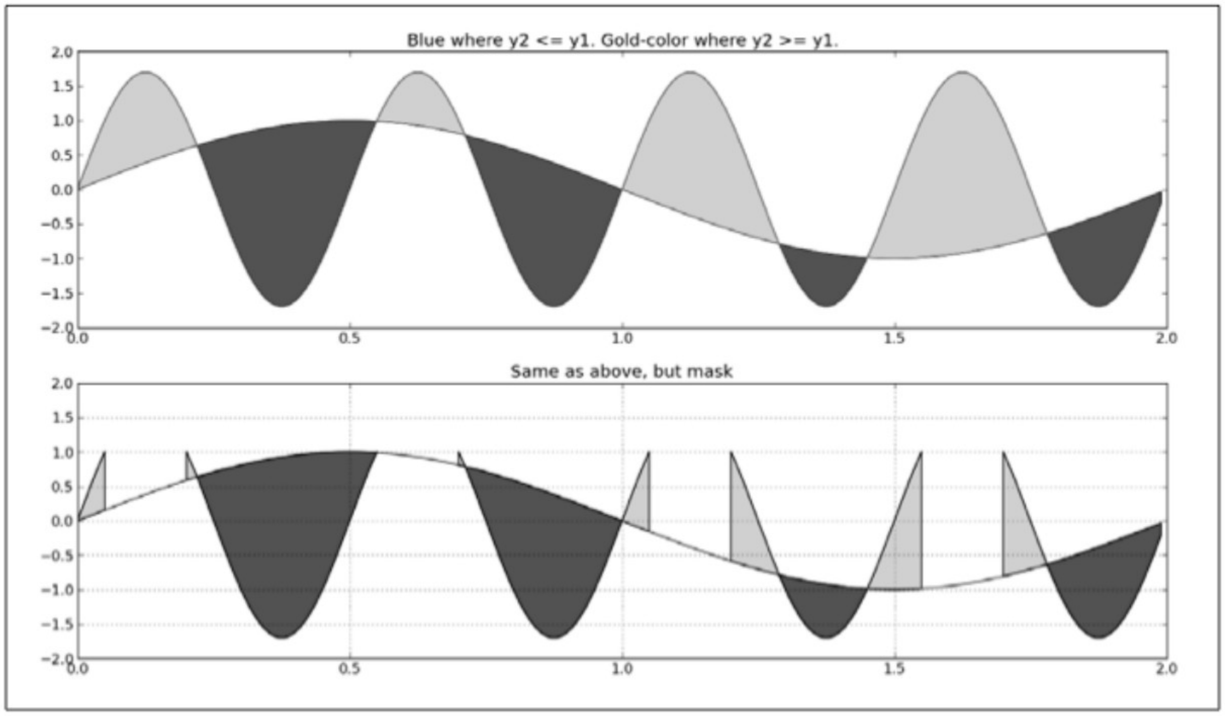
□4- 8

□□□□□□□□□□□□□□□□□□fill□□□□□□□□□□□□□□□□□

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0.0, 2, 0.01)
y1 = np.sin(np.pi*x)
```

```
    y2 = 1.7*np.sin(4*np.pi*x)
    fig = plt.figure()
    axes1 = fig.add_subplot(211)
    axes1.plot(x, y1, x, y2, color='grey')
    axes1.fill_between(x,       y1,       y2,       where=y2<=y1,
facecolor='blue',
    interpolate=True)
    axes1.fill_between(x,       y1,       y2,       where=y2>=y1,
facecolor='gold',
    interpolate=True)
    axes1.set_title('Blue  where  y2  <=  y1.  Gold-color  where
y2 >= y1.')
    axes1.set_ylim(-2,2)
    # Mask values in y2 with value greater than 1.0
    y2 = np.ma.masked_greater(y2, 1.0)
    axes2 = fig.add_subplot(212, sharex=axes1)
    axes2.plot(x, y1, x, y2, color='black')
    axes2.fill_between(x,       y1,       y2,       where=y2<=y1,
facecolor='blue',
    interpolate=True)
    axes2.fill_between(x,       y1,       y2,       where=y2>=y1,
facecolor='gold', interpolate=True)
    axes2.set_title('Same as above, but mask')
    axes2.set_ylim(-2,2)
    axes2.grid('on')
    plt.show()
```

□□□□□□□□□□4-9□□□□□□



[illegible]4-9

4.8.3 [illegible]

[illegible]

[illegible]

[illegible]where[illegible]fill_between()[illegible]where[illegible]True[illegible]where[illegible]N[illegible]

[illegible] mask_greater[illegible]numpy.ma[illegible]

4.9 [illegible]

[illegible]

[illegible]

[illegible] http://www. astronwireless. com/topic-archives-antenna-radiation-patterns.asp[illegible]

[illegible]

4.9.1 [illegible]

[illegible]r[illegible]theta[illegible]matplotlib[illegible]

[illegible] plot()[illegible] polar()[illegible]polar()[illegible]theta[illegible]r[illegible]plot()[illegible]

[illegible] matplotlib [illegible] add_axes [illegible] add_subplot[illegible]polar=True[illegible]

[illegible] matplotlib. pyplot.rgrids()[illegible] matplotlib. pyplot.thetagrid()[illegible]

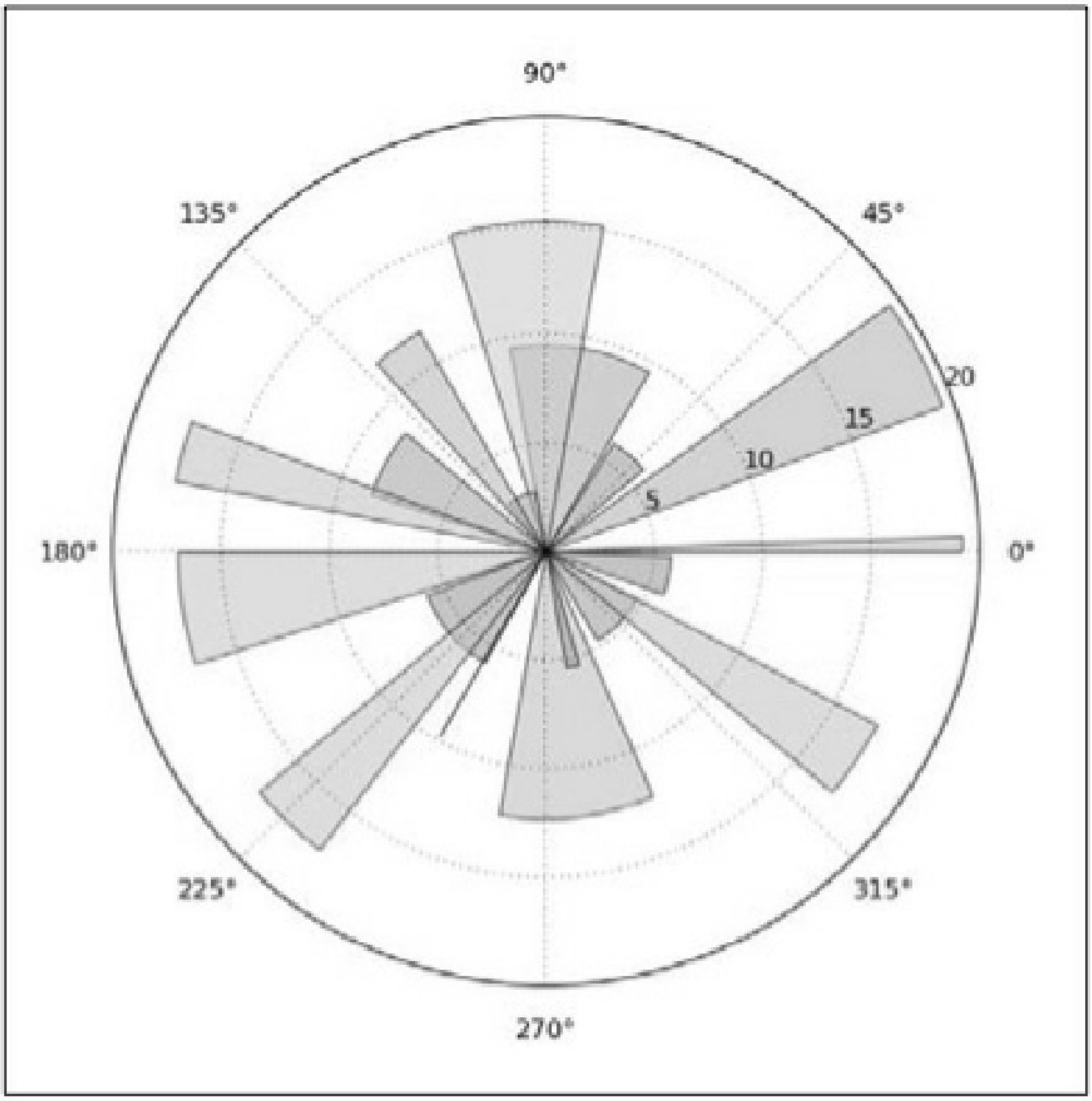
4.9.2 [illegible]

[illegible]

```
import numpy as np
import matplotlib.cm as cm
import matplotlib.pyplot as plt
figsize = 7
colormap = lambda r: cm.Set2(r / 20.)
N = 18 # number of bars
```

```
fig = plt.figure(figsize=(figsize,figsize))
ax = fig.add_axes([0.2, 0.2, 0.7, 0.7], polar=True)
theta = np.arange(0.0, 2 * np.pi, 2 * np.pi/N)
radii = 20 * np.random.rand(N)
width = np.pi / 4 * np.random.rand(N)
bars = ax.bar(theta, radii, width=width, bottom=0.0)
for r, bar in zip(radii, bars):
    bar.set_facecolor(colormap(r))
    bar.set_alpha(0.6)
plt.show()
```

[illegible]4-10[illegible]



[illegible]4-10

4.9.3 [illegible]

[illegible]

[illegible]theta[illegible]radii[illegible]matplotlib.axes.bar[illegible]

matplotlib□□□
bar□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□ax□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

4.10 □□□□□□□□□□□□

□□□□□□□□□□□□□□□□“□□□□”□□□□——□□□matplotlib□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□

4.10.1 □□□□

□□□

□□□Python□matplotlib□□□□□□□□□□□□

4.10.2 □□□□

□□□□□□□□

1.□□□□helper□□□□□□□□□□□□□□□□□□□□□

2.□□□□□□□□draw()□[8]

```
import os
import sys
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np
def build_folders(start_path):
   folders = []
   for each in get_directories(start_path):
      size = get_size(each)
      if size >= 25 * 1024 * 1024:
         folders.append({'size': size, 'path': each})
   for each in folders:
      print "Path: " + os.path.basename(each['path'])
      print "Size: " + str(each['size'] / 1024 / 1024) + " MB"
   return folders
def get_size(path):
   assert path is not None
   total_size = 0
   for dirpath, dirnames, filenames in os.walk(path):
      for f in filenames:
         fp = os.path.join(dirpath, f)
         try:
            size = os.path.getsize(fp)
            total_size += size
            #print "Size of '{0}' is {1}".format(fp, size)
         except OSError as err:
            print str(err)
            pass
   return total_size
def get_directories(path):
   dirs = set()
```

```
        for dirpath, dirnames, filenames in os.walk(path):
            dirs = set([os.path.join(dirpath, x) for x in dirnames])
            break # we just want the first one
        return dirs
    def draw(folders):
        """ Draw folder size for given folder"""
        figsize = (8, 8) # keep the figure square
        ldo, rup = 0.1, 0.8 # leftdown and right up normalized
        fig = plt.figure(figsize=figsize)
        ax = fig.add_axes([ldo, ldo, rup, rup], polar=True)
        # transform data
        x = [os.path.basename(x['path']) for x in folders]
        y = [y['size'] / 1024 / 1024 for y in folders]
        theta = np.arange(0.0, 2 * np.pi, 2 * np.pi / len(x))
        radii = y
        bars = ax.bar(theta, radii)
        middle = 90 / len(x)
        theta_ticks = [t * (180 / np.pi) + middle for t in theta]
        lines,        labels        =        plt.thetagrids(theta_ticks,
labels=x,frac=0.5)
        for step, each in enumerate(labels):
            each.set_rotation(theta[step]   *   (180   /   np.pi)   +
    middle)
            each.set_fontsize(8)
        # configure bars
        colormap = lambda r:cm.Set2(r / len(x))
        for r, each in zip(radii, bars):
```

```
        each.set_facecolor(colormap(r))
        each.set_alpha(0.5)
    plt.show()
```

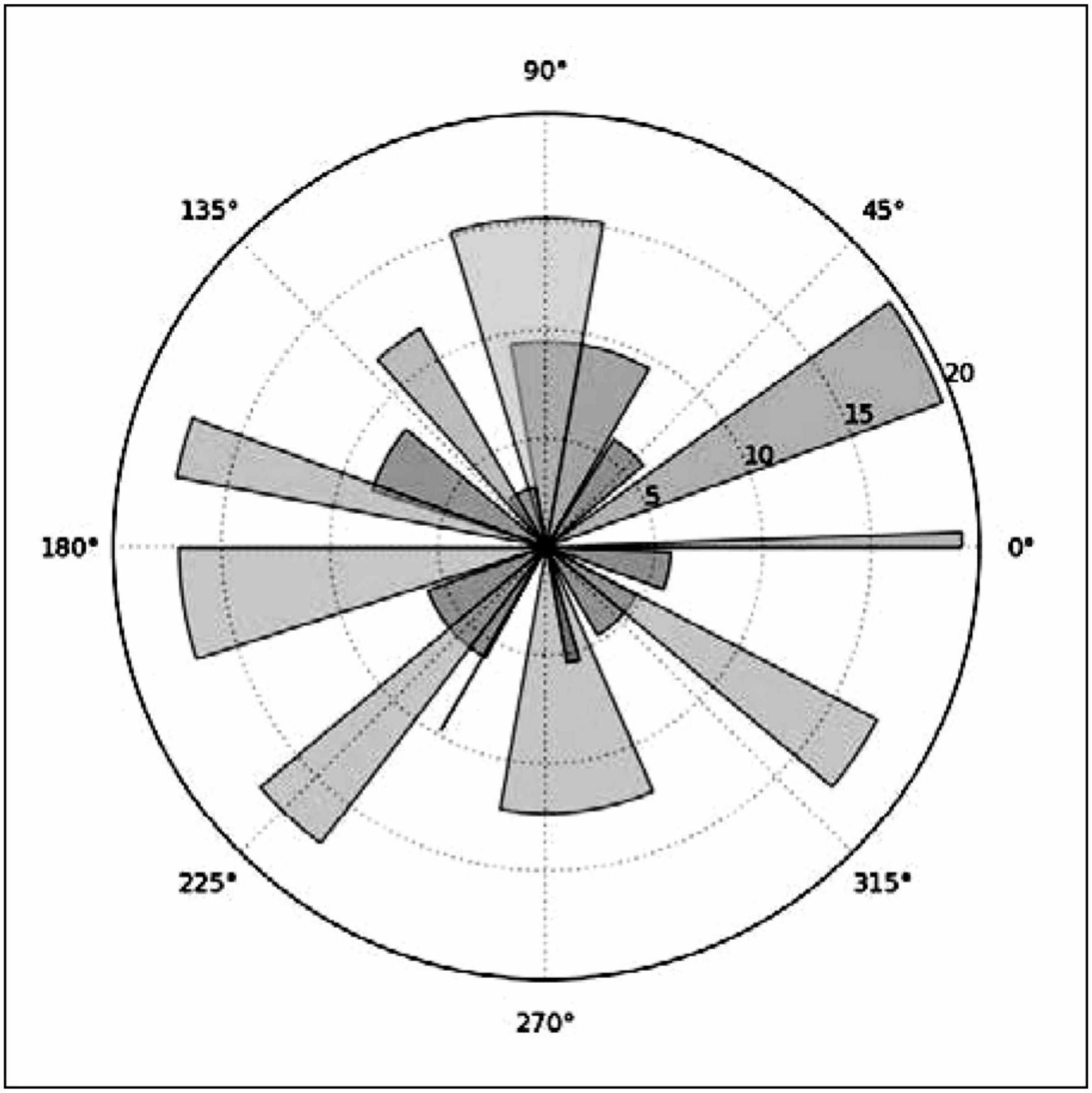
3.[illegible]main[illegible]main[illegible]

```
if __name__ == '__main__':
    if len(sys.argv) is not 2:
        print "ERROR: Please supply path to folder."
        sys.exit(-1)
    start_path = sys.argv[1]
    if not os.path.exists(start_path):
        print "ERROR: Path must exits."
        sys.exit(-1)
    folders = build_folders(start_path)
    if len(folders) < 1:
        print "ERROR: Path does not contain any folders."
        sys.exit(-1)
    draw(folders)
```

[illegible]

```
$ python ch04_rec11_filesystem.py /usr/lib/
```

[illegible]4-11[illegible]



□4-11

4.10.3 □□□□

□□□□□□□ if __name__== '__main__'□□□□□□□□□□□□□□□□□□□□
□□□□□

□□sys□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□ build_folders □□□□□□□□□□□□□□□□□□□□□□□ start_path□□
□□□□□□□□□□□□□get_directories□□start_path□□□□□□□□□□□□□□□
□□□□□□□□□get_size□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□draw□draw□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□

□□

[1]. patch□□

[2]. □□□□□□□□

[3]. □□□ 1/71□□□□□□□□

[4]. □□□ im1□im2□im3□□□□□□□□

[5].□□□□□ z=f(x,y)□x □ y □□□□□□□□□z □□□□□□□□□□ z □□□□□□□□□□□□□□□□□

[6]. □□□□□□ process_signals □□□

[7]. □□□ matplotlib.pyploy.fill_betweenx()□□□□□□□□

[8]. □□□□□ 2 □□□□□□□□□□□□□□□ 3 □□□□□□□□□□□□□

□5□ □□3D□□□□□

□□□□□□□□□□□

◆ □□ 3D □□□

◆ □□ 3D □□□

◆ □ matplotlib □□□□□

◆ □ OpenGL □□□□

5.1 □□

3D□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□

□□□□□□□□□3D□□□□□□

5.2 □□ 3D □□□

□□matplotlib□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□Excel□3D□□□□□□□matplotlib□□□□□□□□□□□□
□toolkits□□□□□□□□□□□□□□3D□□□□□□□□□□

□□□□□□□ Basemap□GTK □□□Excel □□□Natgrid□AxesGrid
□ mplot3d□

□□□□□□□ mplot3d □□□□□□mpl_toolkits.mplot3□□□□□□□□□□□3D□□□□□□□□□□□□□□□□□□□□scatter□□□□□□surf□□□□□line□□□□□□mesh□□□□mplot3d□□□□□□□3D□□□□□□□□□□□□□matplotlib□□□□□□□□□□□□□□□□□□□

5.2.1 □□□□

□□3D□□□□□□□□□□□□Axes3D□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□3□□□□□□□□□

□□□□□□□□□□ mpl_toolkits.mplot3d.Axes3D.plot □□ xs□ys□zs □zdir □□□□□□□□□□□□□ matplotlib.axes.Axes.plot□□□□□□□□□□□□□□□□

1.xs□ys□x□□y□□□□

2.zs□□□z□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

3.zdir□□□□□□□□□□z□□□□□□□□zs□□□□□□□xs□□ys□□

□ □ mpl_toolkits.mplot3d.art3d □ □ □ 3D artist □ □ □ □ 2Dartists□□□3D□□□□□□□□□□□□□□□rotate_axes□□□□□□□□□□□□□Axes3D□□□□□□□□□□□□□□□□□zdir□□□□□□zdir□□□□z□□□□□□□□□'-'□□□□□□□□□□□□□zdir□□□□□□x□-x□y□-y□z□□-z□

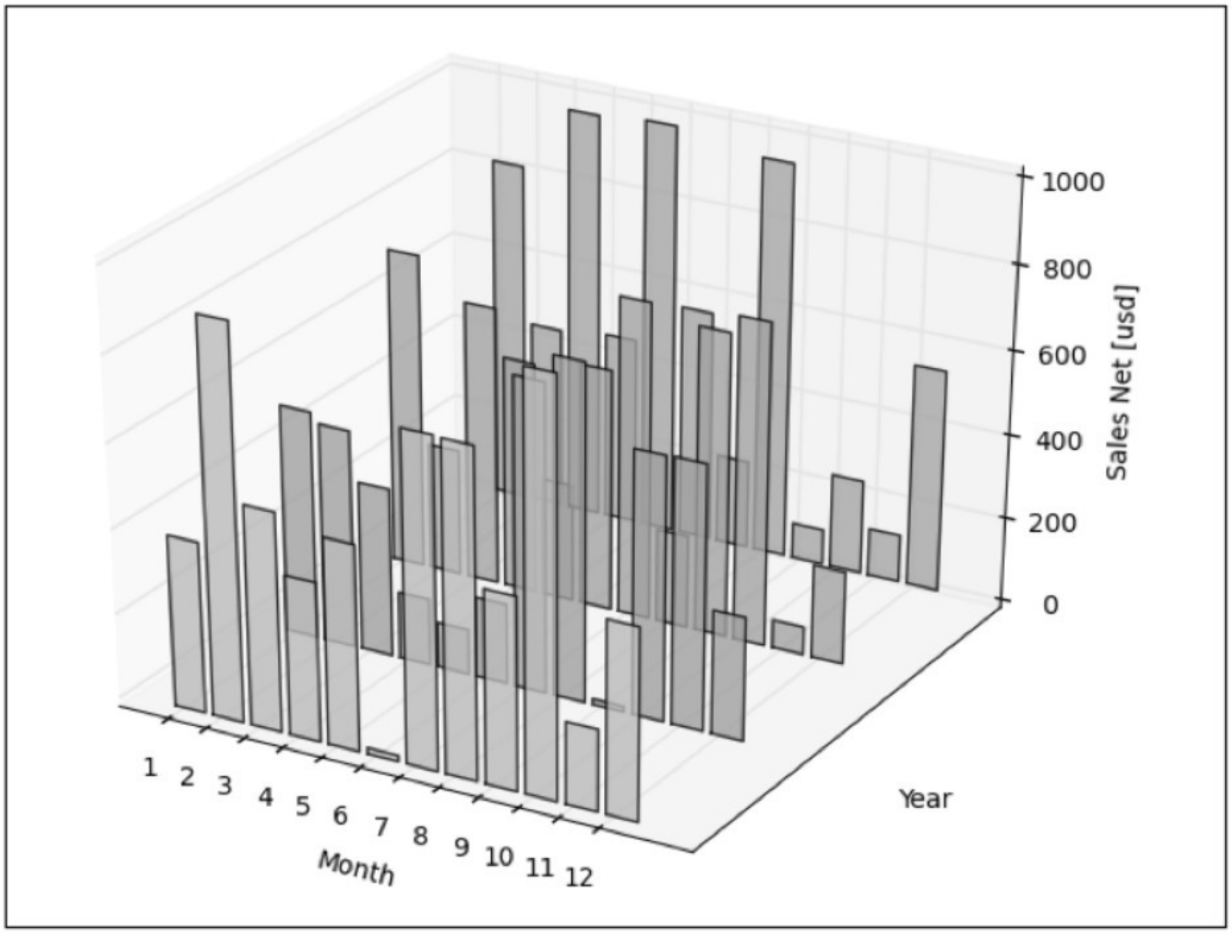
5.2.2 □□□□

□□□□□□□□□□□□□□□□

import random

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from mpl_toolkits.mplot3d import Axes3D
mpl.rcParams['font.size'] = 10
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
for z in [2011, 2012, 2013, 2014]:
    xs = xrange(1,13)
    ys = 1000 * np.random.rand(12)
    color                                                        =
plt.cm.Set2(random.choice(xrange(plt.cm.Set2.N)))
    ax.bar(xs, ys, zs=z, zdir='y', color=color, alpha=0.8)
ax.xaxis.set_major_locator(mpl.ticker.FixedLocator(xs))
ax.yaxis.set_major_locator(mpl.ticker.FixedLocator(ys))
ax.set_xlabel('Month')
ax.set_ylabel('Year')
ax.set_zlabel('Sales Net [usd]')
plt.show()
```

[illegible]5-1[illegible]



□5-1

5.2.3 □□□□

□□□□□□ 2D □□□□□□□□□□□□□□□□□□□□□□□□□□□□□backend□□□□□□□□□□□□□□□□□□4□□□□□□2011-2014□□

□□□□□3D□□□□□□□□Z□□

□□□□□□□□□□□□□□□□□□□□□□□□□Z-order□□□xs□ys□□□□□□□□□□xs□ys□□□□□□□□□□

5.2.4 □□□□

□□□□□matplotlib□2D□□□□□□□□□□□□□□□□scatter()□plot()□□
□□□□□□□□□□□□□□□□□□□□□contour□contourf□bar□□□□□□

□□ 3D □□□□□□□□□□□□□□□wireframe□□□□□□surface□□□□□□
□tri-surface□□

□□□□□□□□□□□□□□□□□□Pringle□□□□□□□□□□□□□□□□□□□□□□□
□hyperbolic paraboloid□□

from mpl_toolkits.mplot3d import Axes3D

from matplotlib import cm

import matplotlib.pyplot as plt

import numpy as np

n_angles = 36

n_radii = 8

# An array of radii

# Does not include radius r=0, this is to eliminate duplicate points

radii = np.linspace(0.125, 1.0, n_radii)

# An array of angles

angles = np.linspace(0, 2 * np.pi, n_angles, endpoint=False)

# Repeat all angles for each radius

angles = np.repeat(angles[..., np.newaxis], n_radii, axis=1)

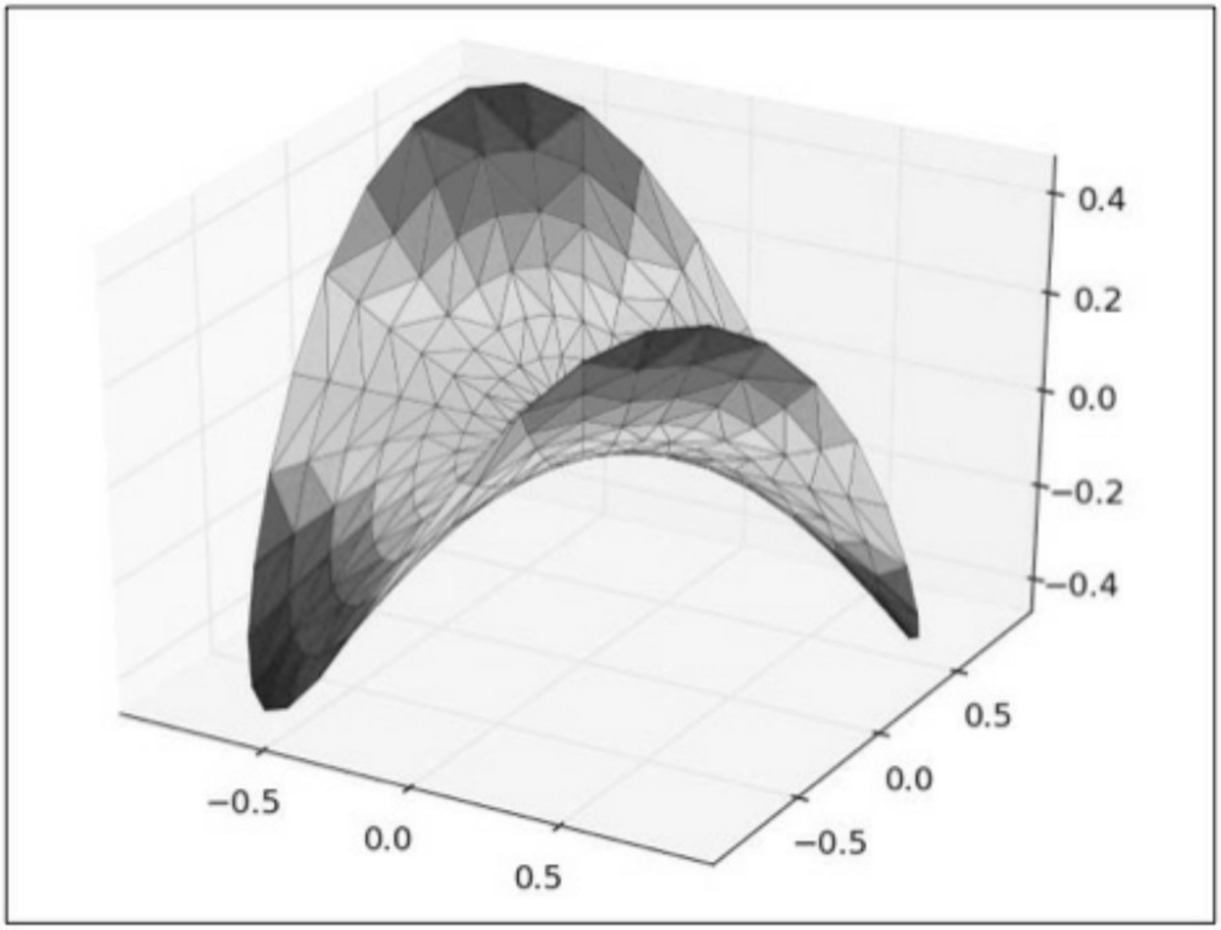
# Convert polar (radii, angles) coords to cartesian (x, y) coords

# (0, 0) is added here. There are no duplicate points in the (x, y)

plane

```
x = np.append(0, (radii * np.cos(angles)).flatten())
y = np.append(0, (radii * np.sin(angles)).flatten())
# Pringle surface
z = np.sin(-x * y)
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot_trisurf(x, y, z, cmap=cm.jet, linewidth=0.2)
plt.show()
```

□□□□□□□□□5-2□□□□□□



□5-2

5.3 [illegible] 3D [illegible]

[illegible]3D[illegible]3D[illegible]3D[illegible]3[illegible]
[illegible]x, y[illegible]

[illegible]3D[illegible]

5.3.1 [illegible]

[illegible]“bin”[illegible]
[illegible]

5.3.2 [illegible]

[illegible]

1.[illegible]Numpy[illegible]

2.[illegible]x[illegible]y[illegible]

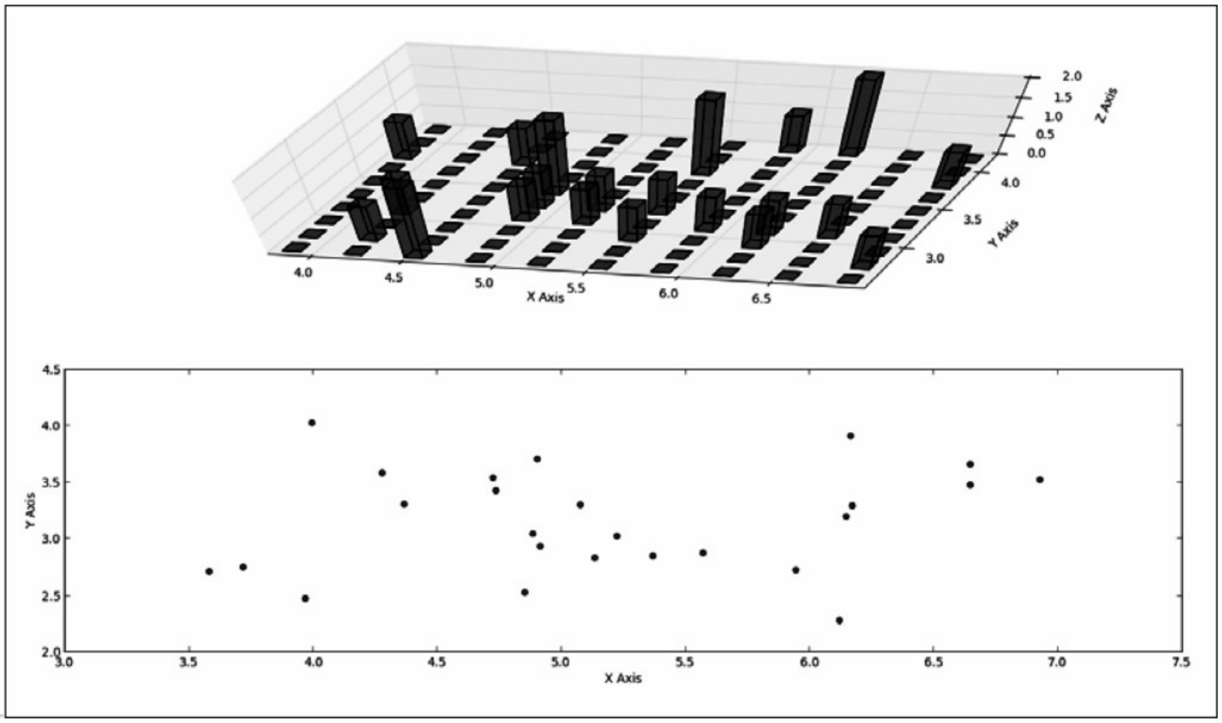
3.[illegible]3D[illegible]

[illegible]

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
mpl.rcParams['font.size'] = 10
samples = 25
x = np.random.normal(5, 1, samples)
y = np.random.normal(3, .5, samples)
fig = plt.figure()
ax = fig.add_subplot(211, projection='3d')
```

```
    # compute two-dimensional histogram
    hist, xedges, yedges = np.histogram2d(x, y, bins=10)
    # compute location of the x,y bar positions
    elements = (len(xedges) - 1) * (len(yedges) - 1)
    xpos,     ypos     =     np.meshgrid(xedges[:-1]+.25,
yedges[:-1]+.25)
    xpos = xpos.flatten()
    ypos = ypos.flatten()
    zpos = np.zeros(elements)
    # make every bar the same width in base
    dx = .1 * np.ones_like(zpos)
    dy = dx.copy()
    # this defines the height of the bar
    dz = hist.flatten()
    ax.bar3d(xpos,  ypos,  zpos,  dx,  dy,  dz,  color='b',
alpha=0.4)
    ax.set_xlabel('X Axis')
    ax.set_ylabel('Y Axis')
    ax.set_zlabel('Z Axis')
    # plot the same x,y correlation in scatter plot
    # for comparison
    ax2 = fig.add_subplot(212)
    ax2.scatter(x, y)
    ax2.set_xlabel('X Axis')
    ax2.set_ylabel('Y Axis')
    plt.show()
```

[illegible]5-3[illegible]



□5-3

5.3.3 □□□□

□□□ np.histogram2d □□□□□□□□□□□□□□□□□□□□hist□□x bin□□□ y bin□□□

bar3d □□□□ x, y □□□□□□□□□□□□□□□□□□□□□□□□np.meshgrid□□□x□y□□□□□□□□2D□□□□□□□□□□□□□□□□□□□□□xy□□□□□□□□□□

□□dx□dy□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□xy□□□□□□□□□□□□□ 0.1 □□□□□□

z□□□□□dz□□□□□□□□□□□□□□□hist□□□□□□□□□□□□bin□□□□x□y□□□□□□

□□□□□□□□□5-3□□□□□□□2D□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□3D□□□3D□□□□2D□□□□□□□□□□□□□□2D□□3D□□□□□□□□□

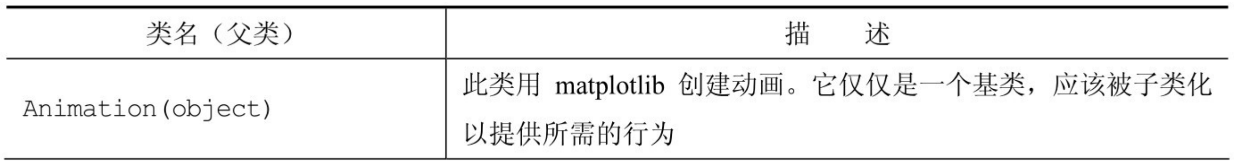
5.4 □matplotlib□□□□□

□□

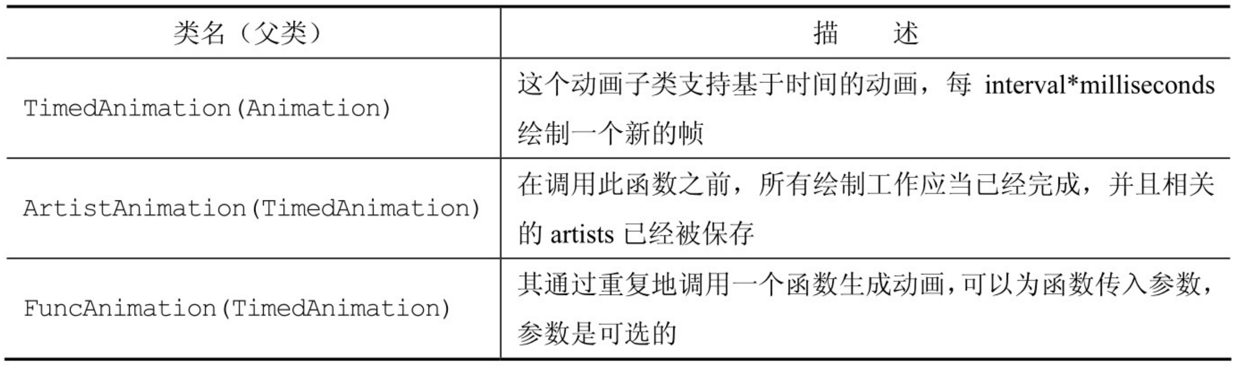
5.4.1 □□□□

□ 1.1 □□□□□□□□□□□□□□□□□□□□ matplotlib □□□□□□□□□□□□matplotlib.animation.Animation□□□TimedAnimation□ArtistAnimation □FuncAnimation□□5-1□□□□□□□□□□□

□5-1



□□



类名（父类）	描　　述
Animation(object)	此类用 matplotlib 创建动画。它仅仅是一个基类，应该被子类化以提供所需的行为
TimedAnimation(Animation)	这个动画子类支持基于时间的动画，每 interval*milliseconds 绘制一个新的帧
ArtistAnimation(TimedAnimation)	在调用此函数之前，所有绘制工作应当已经完成，并且相关的 artists 已经被保存
FuncAnimation(TimedAnimation)	其通过重复地调用一个函数生成动画，可以为函数传入参数，参数是可选的

□□□□□□□□□□□□□□□□□□□□□ffmpeg□□mencoder□□□Google□□□

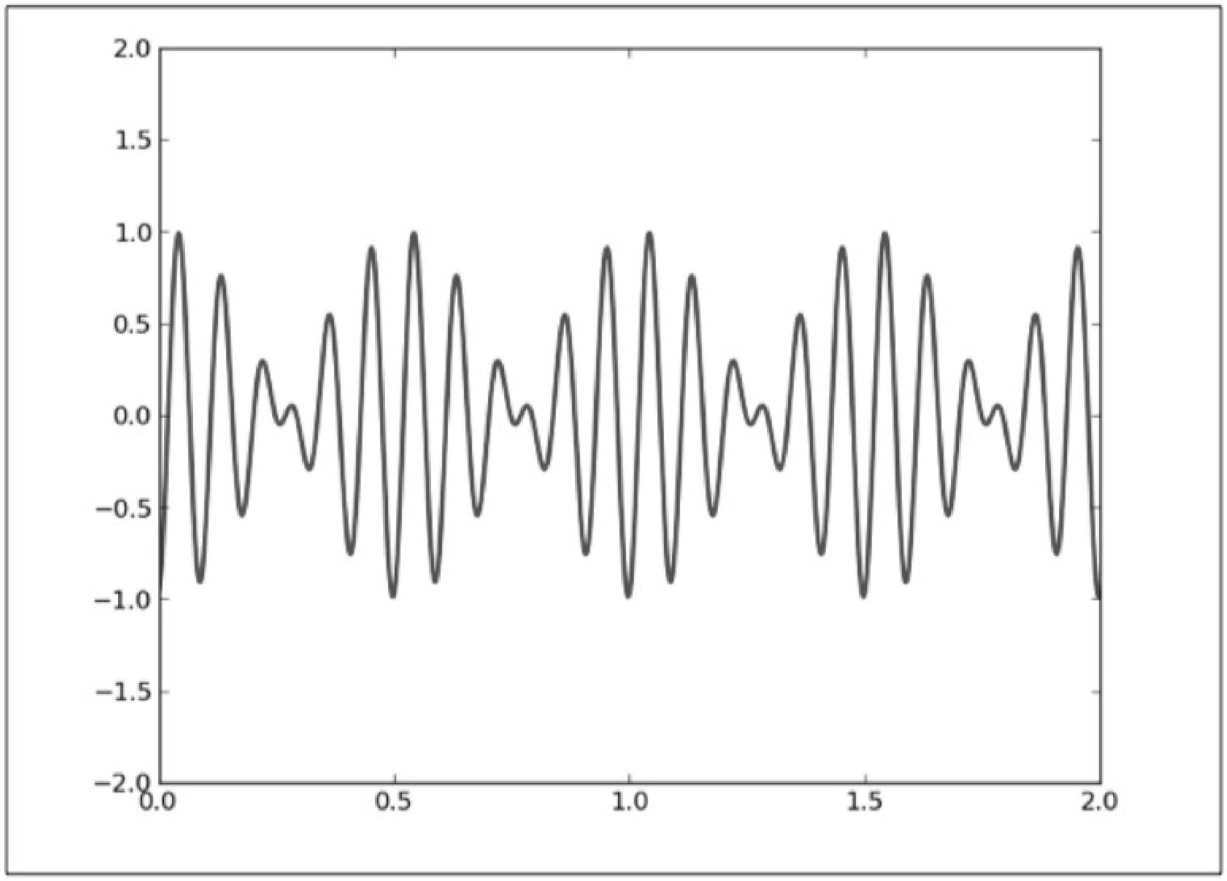
□□□□□□□

5.4.2 □□□□

□□□□□□□□□matplotlib□□□

```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import animation
fig = plt.figure()
ax = plt.axes(xlim=(0, 2), ylim=(-2, 2))
line, = ax.plot([], [], lw=2)
def init():
    """Clears current frame."""
    line.set_data([], [])
    return line,
def animate(i):
    """Draw figure.
    @param i: Frame counter
    @type i: int
    """
    x = np.linspace(0, 2, 1000)
    y = np.sin(2 * np.pi * (x - 0.01 * i)) * np.cos(22 * np.pi * (x - 0.01 * i))
    line.set_data(x, y)
    return line,
# This call puts the work in motion
# connecting init and animate functions and figure we want to draw
animator = animation.FuncAnimation(fig, animate, init_func=init,
        frames=200, interval=20, blit=True)
# This call creates the video file.
# Temporary, every frame is saved as PNG file
# and later processed by ffmpeg encoder into MPEG4 file
# we can pass various arguments to ffmpeg via extra_args
animator.save('basic_animation.mp4', fps=30,
  extra_args=['-vcodec', 'libx264'],
  writer='ffmpeg_file')
plt.show()
```

□□□□□□□□□□□□□□□□□□□ basic_animation.mp4□□□□□□□□□□□□□□□□□□□□□□□□□□□□□MPEG-4□□□□□□□□□□□□□□□□□□□□□5-4□□□

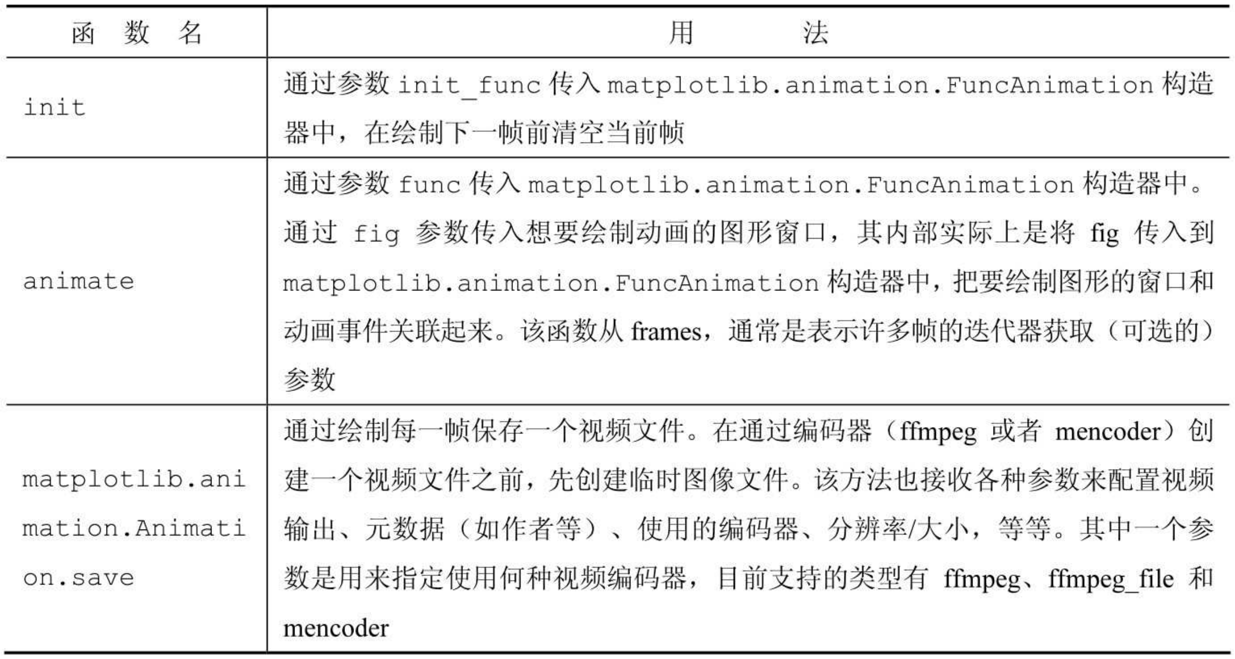


□5- 4

5.4.3 □□□□

□□□□□□□□□□□□□□□ init()□animate()□ save()□□□□□□□ FuncAnimate[1]□□□□□□□□init□animator□□□□□□□save□□□□□□□□□□□5-2□□□□□□□□□□□□□□

□5-2



函 数 名	用 法
init	通过参数 init_func 传入 matplotlib.animation.FuncAnimation 构造器中，在绘制下一帧前清空当前帧
animate	通过参数 func 传入 matplotlib.animation.FuncAnimation 构造器中。通过 fig 参数传入想要绘制动画的图形窗口，其内部实际上是将 fig 传入到 matplotlib.animation.FuncAnimation 构造器中，把要绘制图形的窗口和动画事件关联起来。该函数从 frames，通常是表示许多帧的迭代器获取（可选的）参数
matplotlib.animation.Animation.save	通过绘制每一帧保存一个视频文件。在通过编码器（ffmpeg 或者 mencoder）创建一个视频文件之前，先创建临时图像文件。该方法也接收各种参数来配置视频输出、元数据（如作者等）、使用的编码器、分辨率/大小，等等。其中一个参数是用来指定使用何种视频编码器，目前支持的类型有 ffmpeg、ffmpeg_file 和 mencoder

5.4.4 □□□□

matplotlib.animation.ArtistAnimation□□□□FuncAnimation□□□□□□□□□□□□□artist□□□□□artist□□□□□□□ArtistAnimation□□Artist□□□□matplotlib.animation.TimedAnimation□□□□□□□N□□□□□□□□□□□□□□□□□□

□□□□□ Mac OS X □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□matplotlib□□□□□□□□□□

5.5 □OpenGL□□□□

□□OpenGL□□□□□CPU□□□

□□□□□□□□GPU□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□/□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

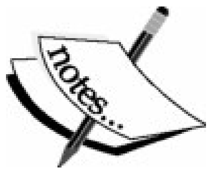
□□□□□□□□□□□□□□□□□“□”□□□OpenGL□□□□□□□□□□□
“□”□

□□□□OpenGL□□□□□□□□□□□□□□□□□□□□□□□□Linux□
Mac □□ Windows □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□

5.5.1 □□□□

□□□□□□□□ OpenGL□□□□□□□□□□□□□□□□□□□□□□□□
OpenGL□□□□□□□□□□□OpenGL□□□□□□□□□□□□□OpenGL□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□NVIDIA
□□AMD/ATI□□□□□□□□□□

□□OpenGL□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□



OpenGL□□□

□□OpenGL□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□/□□□
□□□□□□

□□□□□ OpenGL □□□□□□□□□□ Mac OS X □□OpenGL □□□□□□□□□□□□□□□□□□□□□□□□□“□□□”□□Xcode□□□□□□□□

□Windows□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□OpenGL□□

□ Linux □□□Mesa3D□□□□□□OpenGL□□□□□□□□□□□□□OpenGL□□□□□Xorg□□Linux□FreeBSD□□□□□□□□OpenGL□□□□□

□□□□□Debian/Ubuntu□□□□□□□□□□□□□□□□□□□□

```
$ sudo apt-get install libgl1-mesa-dev libgl-mesa-dri
```

□□□□□□□□□□□□□□□□□□□/□□□□□□□□□□□□OpenGL□□□□□□□□□□

□□□□□□□□□Python□□OpenGL□□□□□□□□□□□□Python□□□□□□□□□□□OpenGL□□□□□□□□□□□□□□□matplotlib□□□□□□□□OpenGL□□□□

◆ Mayavi□□□□□□□□□□□ 3D □□□

◆ Pyglet□□□□□□□ Python □□□□□

◆ Glumpy□□□□□□□□□ Numpy □□□□□□□□□□□□□

◆ Pyglet □ OpenGL□□□□□□□□□□□□□□□□□□□□□□□□□

5.5.2 □□□□

□□□□□□□Mayavi□□□□□□□□□3D□□□□□□□□□□□□□3D□□□□□□□□□□□□□□□Python □□□□ EPD□□□□□□□□□□□□□□□□□□ Windows □ Mac OS X□□□□□□□□□□□□□□□□Linux□□□□□□□□□□pip□□□□□□□□□□□□□

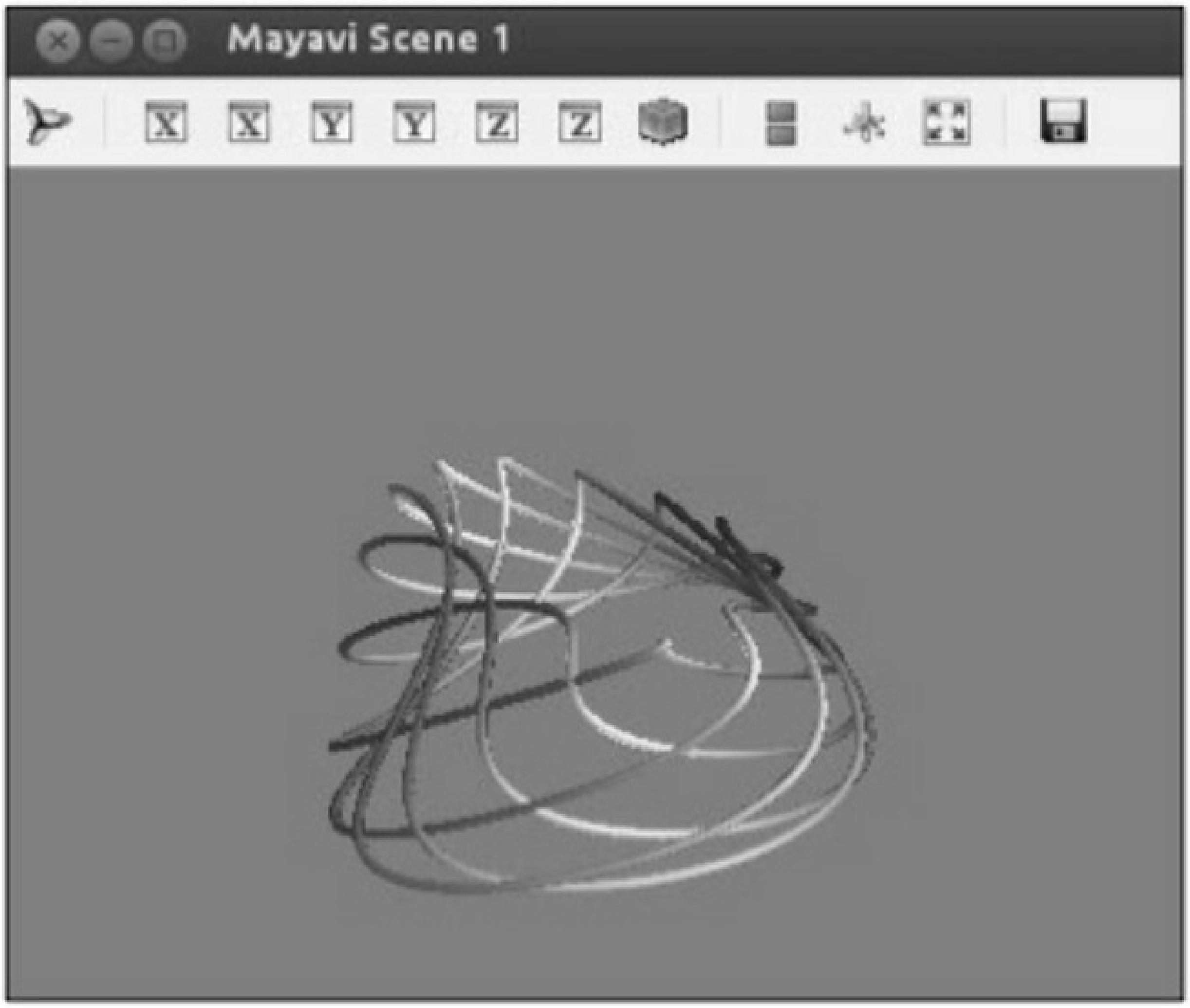
```
$ pip install mayavi
```

Mayavi □□□□□□□□□□□/□□□□□□□□□□□□□□□□□□Mayavi □□□

□□□□□□Mayavi□□□matplotlib□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Mayavi□□□□□mlab□□□□□□□□□□□□□□□□□□□□□□□□□□□Mayavi□□

```
    import numpy
    from mayavi.mlab import *
    # Produce some nice data.
    n_mer, n_long = 6, 11
    pi = numpy.pi
    dphi = pi/1000.0
    phi = numpy.arange(0.0, 2*pi + 0.5*dphi, dphi, 'd')
    mu = phi*n_mer
    x                    =                    numpy.cos(mu)*
(1+numpy.cos(n_long*mu/n_mer)*0.5)
    y                    =                    numpy.sin(mu)*
(1+numpy.cos(n_long*mu/n_mer)*0.5)
    z = numpy.sin(n_long*mu/n_mer)*0.5
    # View it.
    l = plot3d(x, y, z, numpy.sin(mu), tube_radius=0.025,
    colormap='Spectral')
    # Now animate the data.
    ms = l.mlab_source
    for i in range(100):
      x   =   numpy.cos(mu)*(1+numpy.cos(n_long*mu/n_mer
  +
        numpy.pi*(i+1)/5.)*0.5)
      scalars = numpy.sin(mu + numpy.pi*(i+1)/5)
      ms.set(x=x, scalars=scalars)
```

□□□□□□□□□5-5□□□□□□□□□□□□



□5-5

5.5.3 □□□□

□□□□□□□□□□□□□□x□y□z□□□□□□□□□□□□plot3d□□□□□□□□□□□□□□

□□□□□ mlab_source □□□100□□□□□□□

5.5.4 □□□□

[illegible]IPython[illegible]myayvi.lab[illegible]test_*[illegible]

[illegible]IPython[illegible]Python[illegible]

```
In [1]: import mayavi.mlab
In [2]: mayavi.mlab.test_simple_surf??
Type:      function
String Form:<function test_simple_surf at 0x641b410>
File:                                          /usr/lib/python2.7/dist-
packages/mayavi/tools/helper_
functions.py
Definition: mayavi.mlab.test_simple_surf()
Source:
def test_simple_surf():
    """Test Surf with a simple collection of points."""
    x, y = numpy.mgrid[0:3:1,0:3:1]
    return surf(x, y, numpy.asarray(x, 'd'))
```

[illegible]“??”[illegible]IPython[illegible]

Pyglet[illegible]

Pyglet[illegible]Python[illegible]pyglet.gl[illegible]OpenGL[illegible]Pyglet[illegible]pyglet.graphics[illegible]

Pyglet[illegible]Mayavi[illegible]IDE[illegible] OpenGL [illegible] OpenGL [illegible]Mayavi[illegible]

[illegible]

```
import pyglet
window = pyglet.window.Window()
image = pyglet.resource.image('kitten.jpg')
@window.event
def on_draw():
   window.clear()
   image.blit(0, 0)
pyglet.app.run()
```

[illegible] on_draw[illegible]piglet.app.run()[illegible]

[illegible]OpenGL[illegible]pyglet.gl[illegible]pyglet[illegible]pyglet.graphics[illegible]vertex arrays[illegible]buffers[illegible]

Glumpy[illegible]

Glumpy[illegible]OpenGL+NumPy[illegible]OpenGL[illegible]Numpy[illegible] Nicolas Rougier [illegible]Python OpenGL[illegible]bindings[illegible]SciPy[illegible]Glumpy[illegible]

```
sudo apt-get install python-opengl
sudo pip install scipy
sudo pip install glumpy
```

Glumpy [illegible] OpenGL [illegible]textures[illegible]

Pyprocessing [illegible]

Pyprocessing[illegible]Processing[illegible]http://processing.org[illegible]Pyprocessing[illegible]Processing[illegible]Processing[illegible]Python[illegible]Pyprocessing[illegible]

□□□□□□□□□□□□□□pyprocessing□□Pyprocessing□□□□□□□□□□□□□□□□□run()□□□□□

□□□□ OpenGL □□□□ C/C++□□□□□□ binding □□□□□□□□□ OpenGL □□ wiki □□□□□□□□□□□□ http://www.opengl.org/wiki/Getting_started#Tutorials_and_How_To_Guides□

□□□□□□□Python□OpenGL□3D□□□

□□

[1]. □□ FuncAnimation□

[illegible]6[illegible] [illegible]

[illegible]

◆ [illegible] PIL [illegible]

◆ [illegible]

◆ [illegible]

◆ [illegible] Basemap [illegible]

◆ [illegible] Google Map API [illegible]

◆ [illegible] CAPTCHA [illegible]

6.1 [illegible]

[illegible]Python[illegible]

[illegible]PIL[illegible]

[illegible]matplotlib[illegible]annotation[illegible]

[illegible]Python[illegible]API[illegible]

[illegible]Python[illegible]CAPTCHA[illegible]

6.2 [illegible]PIL[illegible]

[illegible] WIMP [illegible]http://en.wikipedia.org/wiki/WIMP_(computing)[illegible]WYSIWYG [illegible]http://en.wikipedia.org/wiki/WYSIWYG[illegible] Python[illegible]

6.2.1 [illegible]

[illegible]PIL[illegible]0[illegible]0[illegible]

Image[illegible]im[illegible]

◆ im = Image.open(filename):[illegible] im[illegible]

◆ im.crop(box): [illegible] box.box[illegible] box = (0, 100, 100, 100)[illegible]

◆ im.filter(filter):[illegible]

◆ im.histogram():[illegible] 256[illegible]RGB[illegible]768[illegible]256[illegible]

◆ im.resize(size, filter):[illegible]resampling[illegible]NEAREST[illegible]BILINEAR[illegible]BICUBIC[illegible] ANTIALIAS[illegible]NEAREST[illegible]

◆ im.rotate(angle, filter):[illegible]

◆ im.split():[illegible]band[illegible]RGB[illegible]3[illegible]

◆ im.transform(size, method, data, filter):[illegible] data [illegible] filter [illegible]AFFINE[illegible]EXTENT[illegible]QUAD[illegible]MESH[illegible]Data[illegible]

ImageDraw□□□□□□□□□□□□□arc□ellipse□pieslice□point□polygon□□□□□□□□□□□□

ImageChops□□□□□□□□□□□□□□□□ Chops□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□8□□□□□□□□□□□□□□□

◆ ImageChops.duplicate(image)□□□□□□□□□□□□□□

◆ ImageChops.invert(image)□□□□□□□□□□□□

◆ ImageChops.difference(image1, image2)□□□□□□□□□□□□□□□□□□□□

ImageFilter □□□□□□□convolution kernels□□□BLUR□MedianFilter□□

ImageFilter □□□

□IPython□□□□□□□□□□□□□□□□□□□□□

```
In [1]: import ImageFilter
In [2]: [f for f in dir(ImageFilter) if f.isupper()]
Out[2]:
['BLUR',
'CONTOUR',
'DETAIL',
'EDGE_ENHANCE',
'EDGE_ENHANCE_MORE',
'EMBOSS',
'FIND_EDGES',
'SHARPEN',
'SMOOTH',
'SMOOTH_MORE']
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
import os
import sys
from PIL import Image, ImageChops, ImageFilter
class DemoPIL(object):
   def __init__(self, image_file=None):
      self.fixed_filters = [ff for ff in dir(ImageFilter) if
ff.isupper()]
      assert image_file is not None
      assert os.path.isfile(image_file) is True
      self.image_file = image_file
      self.image = Image.open(self.image_file)
   def _make_temp_dir(self):
      from tempfile import mkdtemp
      self.ff_tempdir = mkdtemp(prefix="ff_demo")
   def _get_temp_name(self, filter_name):
      name, ext =
os.path.splitext(os.path.basename(self.image_file))
      newimage_file = name + "-" + filter_name + ext
      path = os.path.join(self.ff_tempdir, newimage_file)
      return path
   def _get_filter(self, filter_name):
      note the use python's eval() builtin here to return
function object
```

```
            real_filter = eval("ImageFilter." + filter_name)
            return real_filter
        def apply_filter(self, filter_name):
            print "Applying filter: " + filter_name
            filter_callable = self._get_filter(filter_name)
            # prevent calling non-fixed filters for now
            if filter_name in self.fixed_filters:
                temp_img = self.image.filter(filter_callable)
            else:
                print "Can't apply non-fixed filter now."
            return temp_img
        def run_fixed_filters_demo(self):
            self._make_temp_dir()
            for ffilter in self.fixed_filters:
                temp_img = self.apply_filter(ffilter)
                temp_img.save(self._get_temp_name(ffilter))
            print "Images are in: {0}".format((self.ff_tempdir),)
    if __name__ == "__main__":
        assert len(sys.argv) == 2
        demo_image = sys.argv[1]
        demo = DemoPIL(demo_image)
        # will create set of images in temporary folder
        demo.run_fixed_filters_demo()
```

[illegible]

```
$ python ch06_rec01_01_pil_demo.py image.jpeg
```

[illegible] DemoPIL [illegible] run_fixed_filters_demo[illegible]

[illegible]

[illegible]ImageFilter[illegible]

[illegible] ImageFilter[illegible]

6.2.2 [illegible]

[illegible]0.1[illegible]thumbnail_folder[illegible]

```
import os
import sys
from PIL import Image
class Thumbnailer(object):
    def __init__(self, src_folder=None):
        self.src_folder = src_folder
        self.ratio = .3
        self.thumbnail_folder = "thumbnails"
    def _create_thumbnails_folder(self):
        thumb_path = os.path.join(self.src_folder, self.thumbnail_folder)
        if not os.path.isdir(thumb_path):
            os.makedirs(thumb_path)
    def _build_thumb_path(self, image_path):
        root = os.path.dirname(image_path)
        name, ext =
os.path.splitext(os.path.basename(image_path))
        suffix = ".thumbnail"
        return os.path.join(root, self.thumbnail_folder, name
+ suffix + ext)
    def _load_files(self):
        files = set()
        for each in os.listdir(self.src_folder):
            each = os.path.abspath(self.src_folder + '/' + each)
            if os.path.isfile(each):
                files.add(each)
        return files
    def _thumb_size(self, size):
        return (int(size[0] * self.ratio), int(size[1] * self.ratio))
    def create_thumbnails(self):
        self._create_thumbnails_folder()
        files = self._load_files()
        for each in files:
            print "Processing: " + each
            try:
                img = Image.open(each)
                thumb_size = self._thumb_size(img.size)
                resized = img.resize(thumb_size,
        Image.ANTIALIAS)
                savepath = self._build_thumb_path(each)
                resized.save(savepath)
            except IOError as ex:
                print "Error: " + str(ex)
    if __name__ == "__main__":
        # Usage:
        # ch06_rec01_02_pil_thumbnails.py my_images
        assert len(sys.argv) == 2
        src_folder = sys.argv[1]
        if not os.path.isdir(src_folder):
            print "Error: Path '{0}' does not
    exits.".format((src_folder))
            sys.exit(-1)
        thumbs = Thumbnailer(src_folder)
        # optionally set the name of thumbnail folder inside
*src_folder*.
        thumbs.thumbnail_folder = "THUMBS"
        # define ratio to resize image to
        # 0.1 means the original image will be resized to 10%
of its size
        thumbs.ratio = 0.1
        # will create set of images in temporary folder
        thumbs.create_thumbnails()
```

6.2.3 □□□□

□□□□□ src_folder □□□□□□□□□□□□□□□□□□□□□ Image. open() □□□□□□□□□□□□□ create_thumbnails()□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□IOError□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

[illegible]_load_files()[illegible]

```
for each in os.listdir(self.src_folder):
    if os.path.isfile(each) and os.path.splitext(each) is in ('.jpg', '.png'):
        self._files.add(each)
```

[illegible]

6.2.4 [illegible]

[illegible]PIL[illegible] open()[illegible] save()[illegible] .png [illegible].jpeg[illegible]save()[illegible]

6.3 [illegible]

[illegible]matplotlib[illegible]

6.3.1 [illegible]

[illegible] Bobby Henderson [illegible] The Gospel of the Flying Spaghetti Monster by Bobby Henderson [illegible]

[illegible] Python matplotlib [illegible]

[illegible] ch06 [illegible]

6.3.2 [illegible]

[illegible]

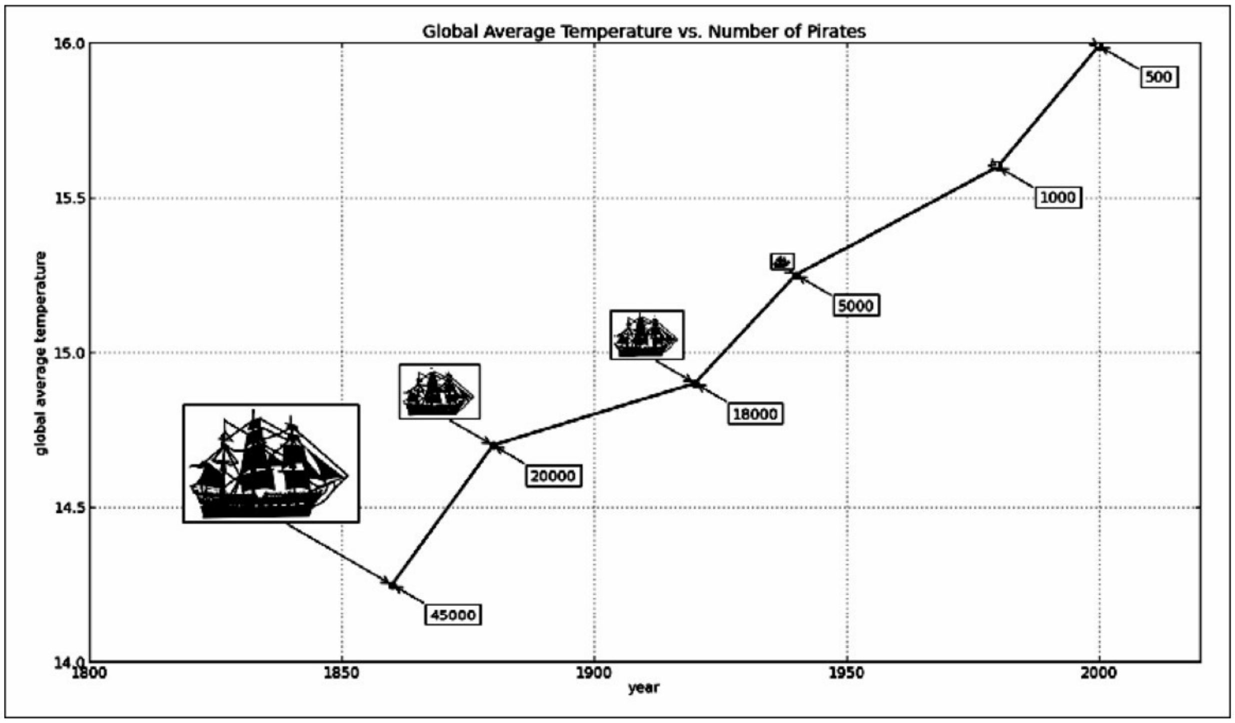
```
import matplotlib.pyplot as plt
from matplotlib._png import read_png
from matplotlib.offsetbox import TextArea, OffsetImage,\
    AnnotationBbox
def load_data():
    import csv
    with open('pirates_temperature.csv', 'r') as f:
        reader = csv.reader(f)
        header = reader.next()
        datarows = []
        for row in reader:
            datarows.append(row)
    return header, datarows
def format_data(datarows):
    years, temps, pirates = [], [], []
    for each in datarows:
        years.append(each[0])
        temps.append(each[1])
        pirates.append(each[2])
    return years, temps, pirates
```

[illegible]helper[illegible]

```
if __name__ == "__main__":
    fig = plt.figure(figsize=(16,8))
    ax = plt.subplot(111) # add sub-plot
header, datarows = load_data()
xlabel, ylabel, _ = header[0]heador[1]
years, temperature, pirates = format_data(datarows)
title = "Global Average Temperature vs. Number of Pirates"
plt.plot(years, temperature, lw=2)
plt.xlabel(xlabel)
plt.ylabel(ylabel)
# for every data point annotate with image and number
for x in xrange(len(years)):
    # current data coordinate
    xy = years[x], temperature[x]
    # add image
    ax.plot(xy[0], xy[1], "ok")
    # load pirate image
    pirate = read_png('tall-ship.png')
    # zoom coefficient (move image with size)
    zoomc = int(pirates[x]) * (1 / 90000.)
    # create OffsetImage
    imagebox = OffsetImage(pirate, zoom=zoomc)
    #   create   anotation   bbox   with   image   and   setup
properties
    ab = AnnotationBbox(imagebox, xy,
      xybox=(-200.*zoomc, 200.*zoomc),
      xycoords='data',
      boxcoords="offset points",
      pad=0.1,
      arrowprops=dict(arrowstyle="->",
        connectionstyle="angle,angleA=0,angleB=-30,rad
    =3")
      )
    ax.add_artist(ab)
    # add text
    no_pirates            =            TextArea(pirates[x],
minimumdescent=False)
    ab = AnnotationBbox(no_pirates, xy,
      xybox=(50., -25.),
      xycoords='data',
      boxcoords="offset points",
      pad=0.3,
      arrowprops=dict(arrowstyle="->",
        connectionstyle="angle,angleA=0,angleB=-30,rad
    =3")
      )
    ax.add_artist(ab)
```

```
plt.grid(1)
plt.xlim(1800, 2020)
plt.ylim(14, 16)
plt.title(title)
plt.show()
```

□□□□□□□□□6-1□□□□□□



□6-1

6.3.3 □□□□

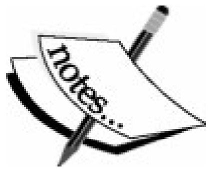
□□□□□□□□□□□□□□□16×8□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ csv □□□□□□□□□□□□□□□ csv reader □□x□□□□□□□□□y□□□□□□□□□□□□□□□□□□□□□

```
xlabel, ylabel, _ = header
```

□□□□□□□□□□□□□□□

plt.xlabel(xlabel)

plt.ylabel(ylabel)



□□□□□□□□□□□□ Python □□□□□□□□□□unpack□□ 3 □□□□□□□□“_” □□□□□□□□□□□□□□□□□□□□□

□load_data□□□header□datarows□□□□□□□□main□

□□□□ format_data()□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□ID□□□

□□□□□x□□□□□□□□□y□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□

□□□□ plot()□□□□□□□/□□□□□□□□□□□□□□□□2 pt□□□□□□□□□□□□ □□□

□□□ □range(len(years))□□□□□□□□□□□□/□□□□□□□□□□□□

ax.plot(xy[0], xy[1], "ok")

□□helper□□read_png□□□□□□□□□□□□□□□□□□□□□

pirate = read_png('tall-ship.png')

□□□□□□□□□□(zoomc)□□□□□□□□□□□□□pirates[x]□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□OffsetImage□□□□□□□AnnotationBbox□□□□□□□□ □□□□□□□□

AnnotationBbox□□□□□□□□□□□□□□□□□□□□□OffsetBox□□□□□□ □Axes.annotate□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□arrowprops□□□□□□□□□□□□□□□□

AnnotateBbox□□□□□□□□□□□

◆ Imagebox[illegible]OffsetBox [illegible]OffsetImage[illegible]

◆ xy:[illegible]

◆ xybox:[illegible]

◆ xycoords:[illegible] xy[illegible]

◆ boxcoords:[illegible] xybox[illegible] xy[illegible]

◆ pad:[illegible]padding[illegible]

◆ arrowprops:[illegible]

[illegible] pirates [illegible]AnnotationBbox[illegible]xybox[illegible]pad[illegible]TextArea[illegible]time.TextArea[illegible]OffsetImage[illegible]OffsetBox[illegible]

[illegible]TextArea[illegible]no_pirates[illegible]AnnotationBbox[illegible]

6.4 [illegible]

[illegible] Python matplotlib [illegible]

6.4.1 [illegible]

[illegible]matplotlib[illegible]imread[illegible]

[illegible]matplotlib[illegible]

6.4.2 [illegible]

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ImageViewer□□□□□□□helper□□□□□□□□

1.□□□□□

2.□□□□□□□□□RGB□□□

3.□□□□□□□□□□□□□

4.□□□□□□□□

5.□□□□□

□□□□□□□□□□□□□□□□□□□□□□□

```
import matplotlib.pyplot as plt
import matplotlib.image as mplimage
import matplotlib as mpl
import os
class ImageViewer(object):
  def __init__(self, imfile):
    self._load_image(imfile)
    self._configure()
    self.figure = plt.gcf()
    t = "Image: {0}".format(os.path.basename(imfile))
    self.figure.suptitle(t, fontsize=20)
    self.shape = (3, 2)
  def _configure(self):
    mpl.rcParams['font.size'] = 10
    mpl.rcParams['figure.autolayout'] = False
    mpl.rcParams['figure.figsize'] = (9, 6)
    mpl.rcParams['figure.subplot.top'] = .9
  def _load_image(self, imfile):
    self.im = mplimage.imread(imfile)
  @staticmethod
  def _get_chno(ch):
    chmap = {'R': 0, 'G': 1, 'B': 2}
    return chmap.get(ch, -1)
  def show_channel(self, ch):
    bins = 256
    ec = 'none'
    chno = self._get_chno(ch)
    loc = (chno, 1)
    ax = plt.subplot2grid(self.shape, loc)
    ax.hist(self.im[:,   :,   chno].flatten(),   bins,   color=ch,
ec=ec,\
            label=ch, alpha=.7)
    ax.set_xlim(0, 255)
    plt.setp(ax.get_xticklabels(), visible=True)
    plt.setp(ax.get_yticklabels(), visible=False)
    plt.setp(ax.get_xticklines(), visible=True)
    plt.setp(ax.get_yticklines(), visible=False)
    plt.legend()
    plt.grid(True, axis='y')
    return ax
  def show(self):
    loc = (0, 0)
    axim = plt.subplot2grid(self.shape, loc, rowspan=3)
    axim.imshow(self.im)
    plt.setp(axim.get_xticklabels(), visible=False)
    plt.setp(axim.get_yticklabels(), visible=False)
    plt.setp(axim.get_xticklines(), visible=False)
    plt.setp(axim.get_yticklines(), visible=False)
    axr = self.show_channel('R')
    axg = self.show_channel('G')
    axb = self.show_channel('B')
    plt.show()
if __name__ == '__main__':
  im = 'images/yellow_flowers.jpg'
  try:
    iv = ImageViewer(im)
    iv.show()
  except Exception as ex:
    print ex
```

6.4.3 [illegible]

[illegible]sys.argv[illegible]im[illegible]

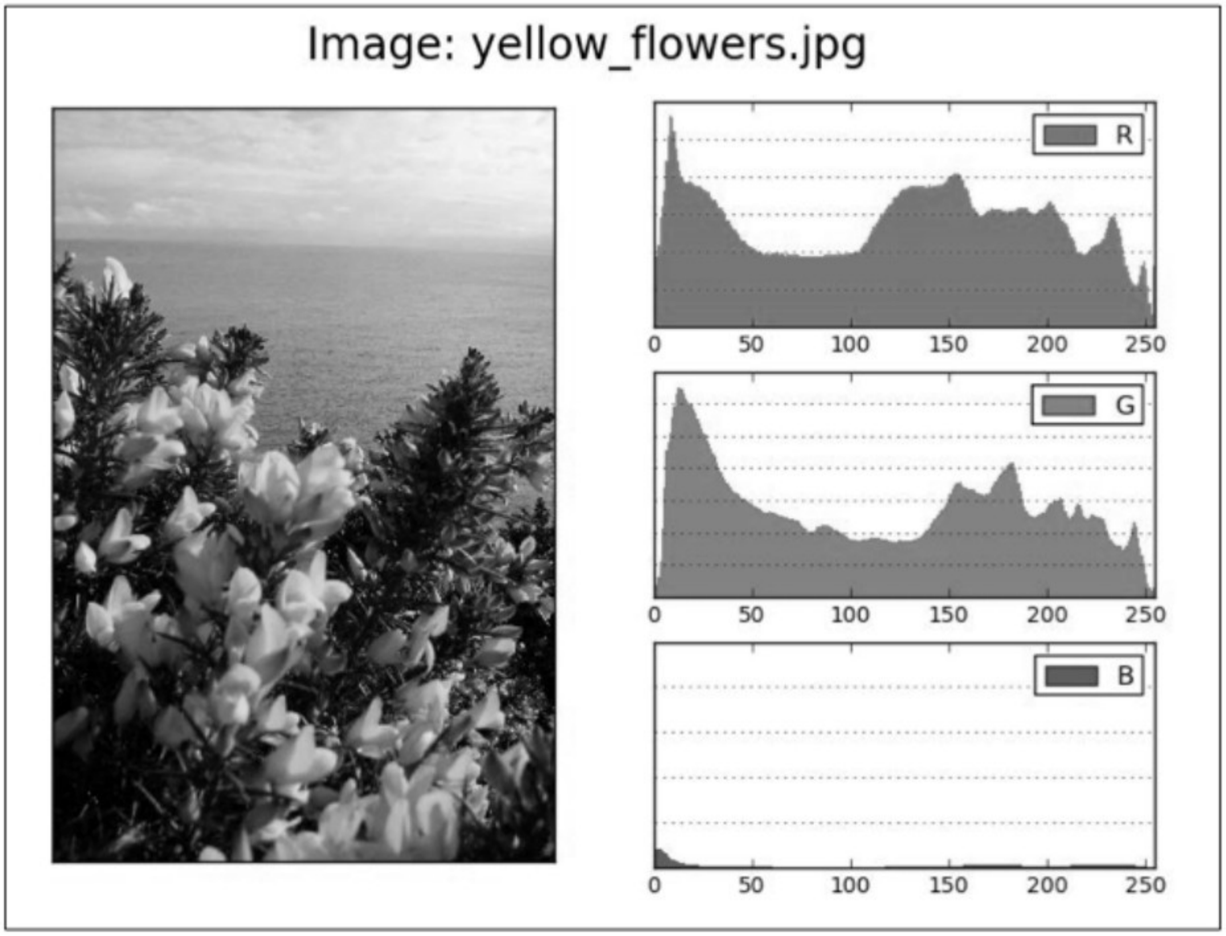
[illegible] ImageViewer[illegible]rcParams[illegible]self.shape[illegible]

[illegible] show()[illegible]

[illegible]show_channel()[illegible]

[illegible]x[illegible]

[illegible]6-2[illegible]



[illegible]6-2

6.4.4 [illegible]

[illegible]matplotlib[illegible]EEG[1][illegible] EEG [illegible]matplotlib.text.Text artists[illegible]

[illegible]matplotlib[illegible]GUI[illegible]matplotlib[illegible]

[illegible]motion_ notify_event[illegible]x[illegible]y[illegible]

6.5 [illegible]Basemap[illegible]

[illegible]

[illegible]matplotlib[illegible]Basemap[illegible]

6.5.1 [illegible]

[illegible]matplotlib[illegible]matplotlib[illegible]Basemap[illegible]

Basemap[illegible]matplotlib[illegible]

[illegible]Basemap[illegible]EPD[illegible]Basemap[illegible]Linux[illegible]Basemap[illegible]Ubuntu[illegible]python-mpltoolkits.basemap[illegible]

$ sudo apt-get install python-mpltoolkits.basemap

[illegible] Mac OS X [illegible] Homebrew[illegible]Fink [illegible] pip [illegible]EPD[illegible]

6.5.2 [illegible]

[illegible]Basemap[illegible]long[illegible]lat[illegible]Mercator projection[illegible]

1.[illegible]Basemap[illegible]merc[illegible]Mercator[illegible]

2.[illegible]Basemap[illegible]

3.[illegible]Basemap[illegible]

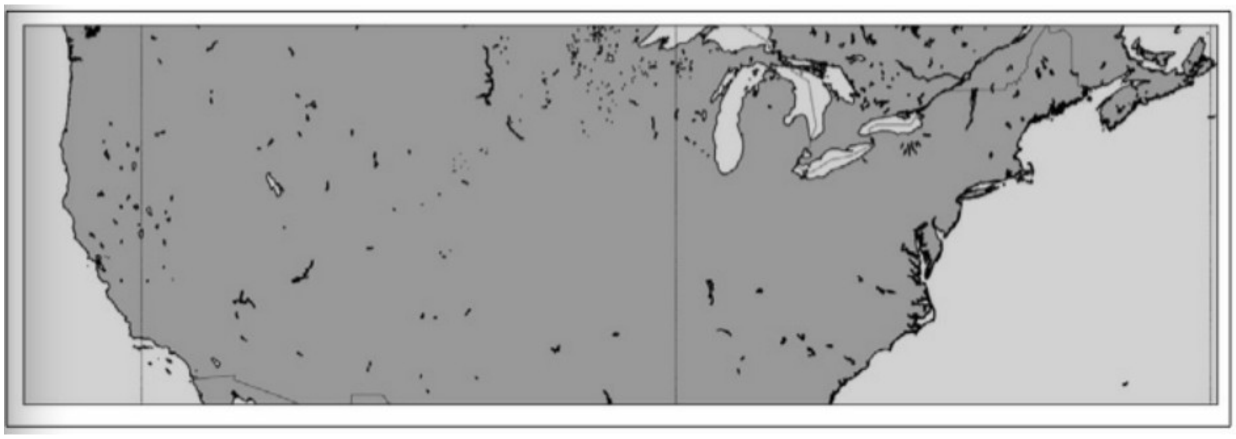
4.[illegible]Basemap[illegible]

5.[illegible]Basemap[illegible]

[illegible]Basemap[illegible]

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
map = Basemap(projection='merc',
      resolution = 'h',
      area_thresh = 0.1,
   llcrnrlon=-126.619875, llcrnrlat=31.354158,
   urcrnrlon=-59.647219, urcrnrlat=47.517613)
map.drawcoastlines()
map.drawcountries()
map.fillcontinents(color='coral', lake_color='aqua')
map.drawmapboundary(fill_color='aqua')
map.drawmeridians(np.arange(0, 360, 30))
map.drawparallels(np.arange(-90, 90, 30))
plt.show()
```

[illegible]6-3[illegible]



□6-3

□□Basemap□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□long/lat□□□□□□□□□□Basemap□□□□□□□matplotlib□□□□□□□□□□□□□cities.shp□cities.shx□□□□□□□□□□□□□□□□□□□□□□□□□ ch06 □□□□□□□□□□□□□□□□

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
map = Basemap(projection='merc',
    resolution = 'h',
    area_thresh = 100,
  llcrnrlon=-126.619875, llcrnrlat=25,
  urcrnrlon=-59.647219, urcrnrlat=55)
shapeinfo = map.readshapefile('cities','cities')
x, y = zip(*map.cities)
# build a list of US cities
city_names = []
for each in map.cities_info:
  if each['COUNTRY'] != 'US':
        city_names.append("")
    else:
        city_names.append(each['NAME'])
map.drawcoastlines()
map.drawcountries()
map.fillcontinents(color='coral', lake_color='aqua')
map.drawmapboundary(fill_color='aqua')
map.drawmeridians(np.arange(0, 360, 30))
map.drawparallels(np.arange(-90, 90, 30))
# draw city markers
map.scatter(x,y,25, marker='o',zorder=10)
# plot labels at City coords.
for city_label, city_x, city_y in zip(city_names, x, y):
    plt.text(city_x, city_y, city_label)
plt.title('Cities in USA')
plt.show()
```

6.5.3 [illegible]

Basemap[illegible]Basemap[illegible]

[illegible]matplotlib.pyplot.show()[illegible]

Basemap[illegible] 32 [illegible]

□□□□Basemap□□□□□□□□□□□□□□□□□□□□□□

```
In [5]: import mpl_toolkits.basemap
In [6]: print mpl_toolkits.basemap.
projections
mbtfpq      McBryde-Thomas Flat-Polar Quartic
aeqd         Azimuthal Equidistant
sinu         Sinusoidal
poly         Polyconic
omerc         Oblique Mercator
gnom         Gnomonic
moll         Mollweide
lcc         Lambert Conformal
tmerc         Transverse Mercator
nplaea      North-Polar Lambert Azimuthal
gall         Gall Stereographic Cylindrical
North-Polar Azimuthal Equidistantnpaeqd
mill         Miller Cylindrical
merc         Mercator
stere         Stereographic
eqdc         Equidistant Conic
cyl         Cylindrical Equidistant
npstere      North-Polar Stereographic
spstere      South-Polar Stereographic
hammer      Hammer
geos         Geostationary
nsper         Near-Sided Perspective
eck4         Eckert IV
```

aea	Albers Equal Area
kav7	Kavrayskiy VII
spaeqd	South-Polar Azimuthal Equidistant
ortho	Orthographic
cass	Cassini-Soldner
vandg	van der Grinten
laea	Lambert Azimuthal Equal Area
splaea	South-Polar Lambert Azimuthal
robin	Robinson

[illegible]

[illegible]

[illegible]

◆ llcrnrlon[illegible]

◆ llcrnrlat[illegible]

◆ urcrnrlon[illegible]

◆ urcrnrlat[illegible]

6.5.4 [illegible]

[illegible]Basemap[illegible]http://matplotlib.org/basemap/users/examples.html[illegible]

[illegible] Basemap [illegible] NetCDF [illegible]NetCDF [illegible]

[illegible]

6.6 [illegible]Google Map API[illegible]

[illegible] Web [illegible] Web[illegible]Python[illegible]HTML[illegible]CSS[illegible]JavaScript[illegible]Python[illegible]Web[illegible]JavaScript[illegible]HTML[illegible]

6.6.1 [illegible]

[illegible] Python [illegible] Google [illegible]Google[illegible]API[illegible]

[illegible] google-visuallization-python [illegible] https://code.google.com/p/google-visualization-python/downloads/detail?name=gviz_api_py-1.8.2.tar.gz&can=2&q=[illegible]

```
$ tar xfv gviz_api_py-1.8.2.tar.gz
$ cd gviz_api_py
$ sudo python ./setup.py install
```

[illegible] Windows [illegible] Mac OS X [illegible] tar.gz[illegible][2] [illegible]

[illegible]virtualenv[illegible]1[illegible]“[illegible]”[illegible]virtualenv[illegible]

[illegible]Web[illegible]Google[illegible]

[illegible]Web[illegible]JavaScript[illegible]

[illegible]Python[illegible]Google[illegible]JavaScript[illegible]Web[illegible]

6.6.2 [illegible]

[illegible]Python[illegible]gdata_viz[illegible]CSV[illegible]Google Geochart [illegible] Table Visualization [illegible]

1.[illegible]

2.[illegible]csv[illegible]CSV[illegible]

3.[illegible]DataTable[illegible]LoadData[illegible]Python[illegible]

4.[illegible]Web[illegible]

[illegible]

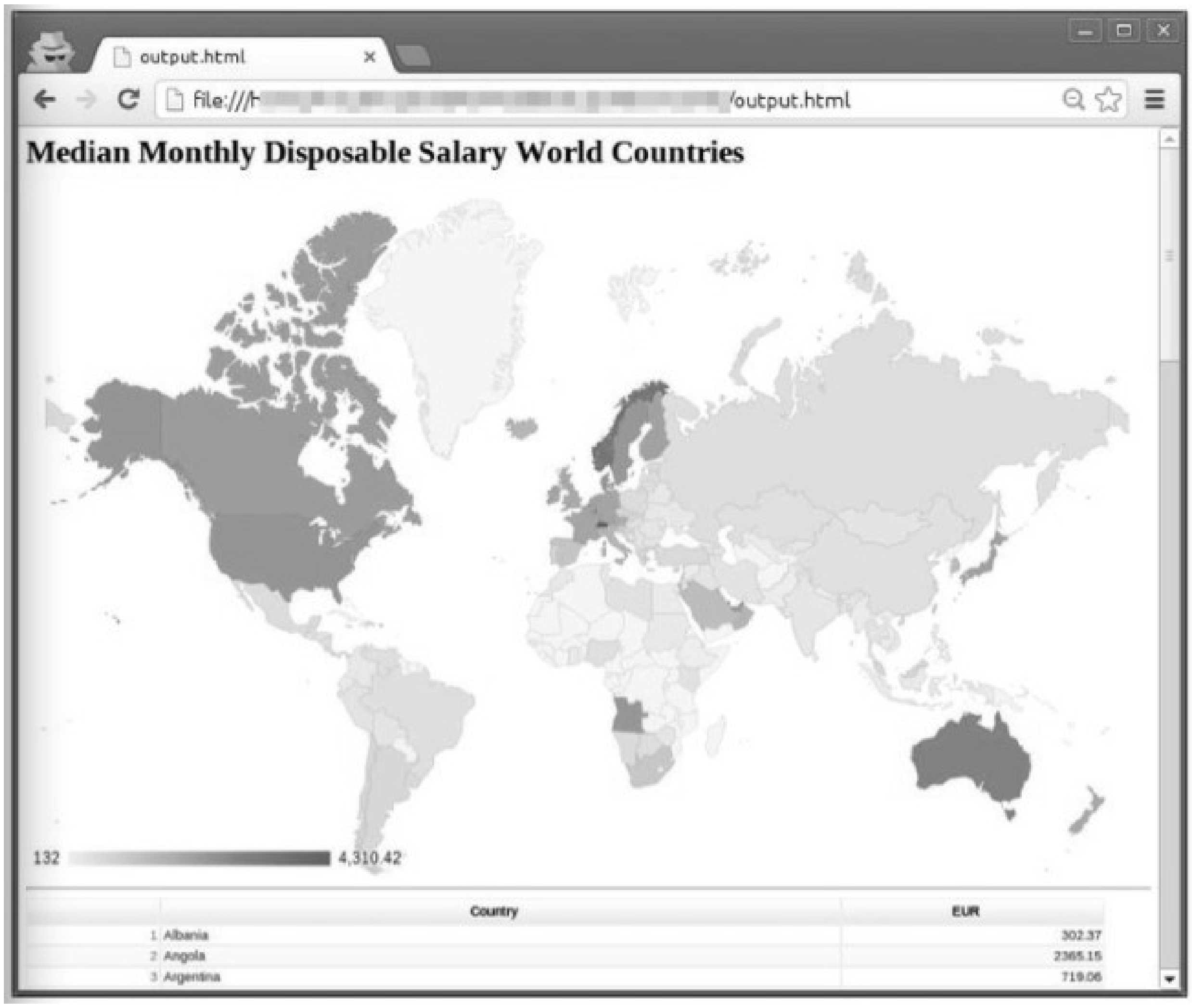
```
import csv
import gviz_api
def get_page_template():
    page_template = """
    <html>
    <script src="https://www.google.com/jsapi" type="text/javascript"> </script>
    <script>
    google.load('visualization', '1', {packages: ['geochart','table']});
    google.setOnLoadCallback(drawMap);
    function drawMap() {
        var json_data = new
    google.visualization.DataTable(%s,0.6);
        var options = {colorAxis: {colors: ['#eee',
    'green']}};
        var mymap = new google.visualization.GeoChart(
          document.getElementById('map_div'));
        mymap.draw(json_data, options);
        var mytable = new google.visualization.Table(
          document.getElementById('table_div'));
        mytable.draw(json_data, {showRowNumber: true})
      }
      </script>
      <body>
      <H1>Median Monthly Disposable Salary World
Countries</H1>
      <div id="map_div"></div>
      <hr />
      <div id="table_div"></div>
      <div id="source">
      <hr />
      <small>
      Source:
      <a href="http://www.numbeo.com/cost-of-
living/prices_by_
    country.jsp? displayCurrency=EUR&itemId=105">
      http://www.numbeo.com/cost-of-
living/prices_by_country.jsp?dis
```

```
play Currency=EUR&itemId=105
   </a>
   </small>
   </div>
   </body>
   </html>
   """
   return page_template
def main():
   # Load data from CVS file
   afile = "median-dpi-countries.csv"
   datarows = []
   with open(afile, 'r') as f:
      reader = csv.reader(f)
      reader.next() # skip header
      for row in reader:
         datarows.append(row)
```

```
   # Describe data
   description = {"country": ("string", "Country"),
         "dpi": ("number", "EUR"), }
   # Build list of dictionaries from loaded data
   data = []
   for each in datarows:
      data.append({"country": each[0],
         "dpi": (float(each[1]), each[1])})
```

```
    # Instantiate DataTable with structure defined in
  'description'
    data_table = gviz_api.DataTable(description)
    # Load it into gviz_api.DataTable
    data_table.LoadData(data)
    # Creating a JSon string
    json = data_table.ToJSon(columns_order=("country",
  "dpi"),
        order_by="country", )
    # Put JSON string into the template
    # and save to output.html
    with open('output.html', 'w') as out:
      out.write(get_page_template() % (json,))
  if __name__ == '__main__':
    main()
```

□□□□□□□output.html□□□□□□□□□□□Web□□□□□□□□□□□□□6-4
□□□



Median Monthly Disposable Salary World Countries

132 4,310.42

	Country	EUR
1	Albania	302.37
2	Angola	2365.15
3	Argentina	719.06

[illegible]6-4

6.6.3 [illegible]

[illegible]main()[illegible]csv[illegible]www.numbeo.com[illegible]CSV[illegible]ch06[illegible]Google[illegible]Python[illegible]ID[illegible]

{"name": ("data_type", "Label")}:

description = {"country": ("string", "Country"),
  "dpi": ("number", "EUR"), }

[illegible]CSV[illegible]data[illegible]

[illegible]gviz_data.DataTable[illegible]data_table[illegible]JSON[illegible]page_template[illegible]

get_page_template()[illegible]HTML[illegible] Google [illegible] Google [illegible]JavaScript [illegible] Google [illegible] JavaScript API [illegible]

```
<script src="https://www.google.com/jsapi"
   type="text/javascript"></script>
```

[illegible]<script>...</script>[illegible]Google[illegible]——geochart[illegible]table[illegible]

```
google.load('visualization', '1', {packages:
['geochart','table']});
```

[illegible]Web[illegible]onLoad[illegible]setOnLoadCallback[illegible]

```
google.setOnLoadCallback(drawMap);
```

[illegible]google[illegible]drawMap()[illegible]drawMap[illegible]JSON[illegible]DataTable[illegible]JavaScript[illegible]

```
var json_data = new google.visualization.DataTable(%s,
0.6);
```

[illegible]ID[illegible]map_div[illegible]HTML[illegible]geochart[illegible]

```
var mymap = new google.visualization.GeoChart(
   document.getElementById('map_div'));
```

[illegible]json_data[illegible]options[illegible]

```
mymap.draw(json_data, options);
```

[illegible]Google[illegible]JavaScript[illegible]

```
var mytable = new google.visualization.Table(
   document.getElementById('table_div'));
```

mytable.draw(json_data, {showRowNumber: true})

[illegible]HTML[illegible]Web[illegible]Python[illegible]HTTP[illegible]Web[illegible]JSON[illegible]

[illegible]HTTP[illegible]http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Response_message [illegible] HTTP[illegible]

[illegible]ToJson()[illegible]ToJsonResponse()[illegible]payload[illegible]JavaScript[illegible]JSON[illegible]data_table[illegible]HTTP[illegible]

6.6.4 [illegible]

[illegible]Python[illegible]HTML/JavaScript/CSS[illegible]Web[illegible]matplotlib[illegible]matplotlib[illegible]

[illegible]Web[illegible]Python[illegible]Web[illegible]Web[illegible]Web[illegible]

[illegible] Google [illegible] https://developers.google.com/chart/interactive/docs/dev/gviz_api_lib[illegible]

6.7 [illegible]CAPTCHA[illegible]

[illegible]Python[illegible]

[illegible]——CAPTCHA [3] [illegible]

6.7.1 [illegible]

CAPTCHA [illegible]Completely Automated Public Turing test to tell Computers and Humans Apart[illegible] Web [illegible]

CAPTCHA [illegible]

[illegible]Python[illegible]

6.7.2 [illegible]

[illegible]CAPTCHA[illegible]

1.[illegible]CAPTCHA[illegible]

2.[illegible]

3.[illegible]

4.[illegible]

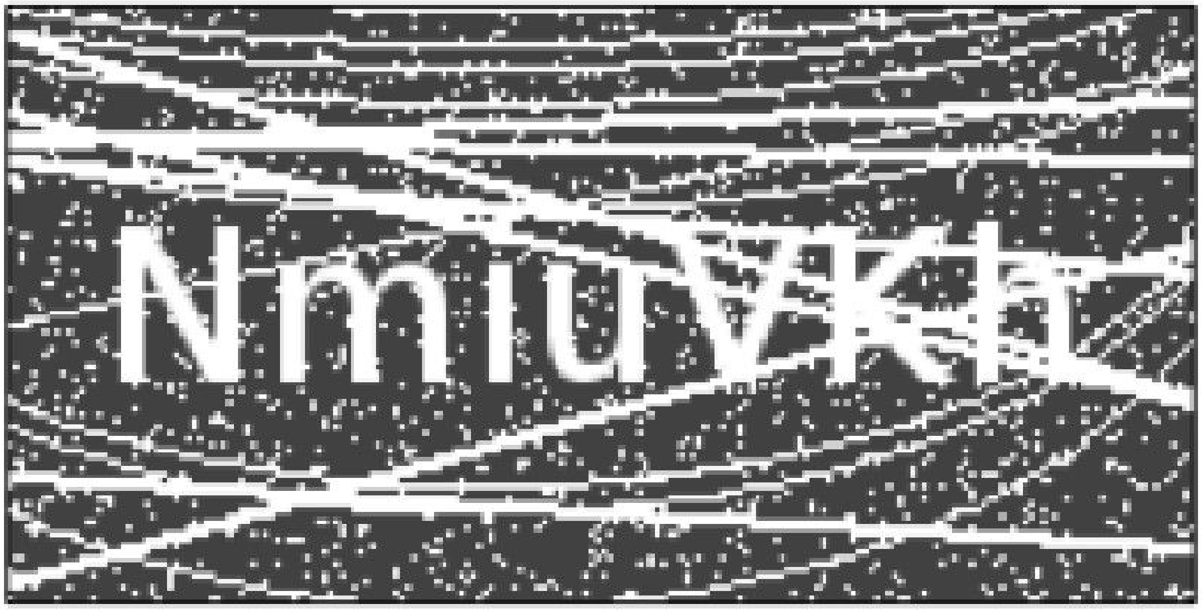
5.[illegible]CAPTCHA[illegible]

6.[illegible]

[illegible]CAPTCHA[illegible]

```
from PIL import Image, ImageDraw, ImageFont
import random
import string
class SimpleCaptchaException(Exception):
  pass
class SimpleCaptcha(object):
  def  __init__(self,  length=5,  size=(200,  100),
fontsize=36,
      random_text=None, random_bgcolor=None):
    self.size = size
    self.text = "CAPTCHA"
    self.fontsize = fontsize
    self.bgcolor = 255
    self.length = length
    self.image = None # current captcha image
    if random_text:
      self.text = self._random_text()
    if not self.text:
raise  SimpleCaptchaException("Field  text  must  not
beempty.")
  if not self.size:
    raise  SimpleCaptchaException("Size  must  not  be
empty.")
  if not self.fontsize:
    raise  SimpleCaptchaException("Font  size  must  be
defined.")
  if random_bgcolor:
    self.bgcolor = self._random_color()
  def _center_coords(self, draw, font):
    width, height = draw.textsize(self.text, font)
    xy = (self.size[0] - width) / 2., (self.size[1] - height) /
2.
    return xy
  def _add_noise_dots(self, draw):
    size = self.image.size
    for _ in range(int(size[0] * size[1] * 0.1)):
      draw.point((random.randint(0, size[0]),
        random.randint(0, size[1])),
        fill="white")
    return draw
  def _add_noise_lines(self, draw):
    size = self.image.size
  for _ in range(8):
    width = random.randint(1, 2)
    start = (0, random.randint(0, size[1] - 1))
    end = (size[0], random.randint(0, size[1] - 1))
    draw.line([start, end], fill="white", width=width)
  for _ in range(8):
    start = (-50, -50)
    end = (size[0] + 10, random.randint(0, size[1] + 10))
    draw.arc(start + end, 0, 360, fill="white")
  return draw
def get_captcha(self, size=None, text=None,
bgcolor=None):
  if text is not None:
    self.text = text
    if size is not None:
        self.size = size
    if bgcolor is not None:
        self.bgcolor = bgcolor
    self.image = Image.new('RGB', self.size, self.bgcolor)
    # Note that the font file must be present
    # or point to your OS's system font
    # Ex. on Mac the path should be
'/Library/Fonts/Tahoma.ttf'
    font = ImageFont.truetype('fonts/Vera.ttf', self.fontsize)
    draw = ImageDraw.Draw(self.image)
    xy = self._center_coords(draw, font)
    draw.text(xy=xy, text=self.text, font=font)
    # Add some dot noise
    draw = self._add_noise_dots(draw)
    # Add some random lines
    draw = self._add_noise_lines(draw)
    self.image.show()
    return self.image, self.text
def _random_text(self):
    letters = string.ascii_lowercase +
string.ascii_uppercase
    random_text = ""
    for _ in range(self.length):
        random_text += random.choice(letters)
    return random_text
def _random_color(self):
    r = random.randint(0, 255)
    g = random.randint(0, 255)
    b = random.randint(0, 255)
    return (r, g, b)
if __name__ == "__main__":
    sc      =      SimpleCaptcha(length=7,      fontsize=36,
random_text=True,
random_ bgcolor=True)
    sc.get_captcha()
```

□□□□□□□□□6-5□□□□□□

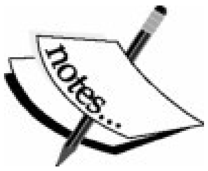


□6-5

6.7.3 □□□□

□□□□□□□□□□□ Python □□□□□□□□□□□□□□□□□□□□□CAPTCHA□□□□□□□

□□□□□□□□□□SimpleCaptcha□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□SimpleCaptchaException□□□□□□□□□□□



□□□Python□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□main□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□sc□□□□get_captcha□□□□□□□□□□□□get_captcha□□Web□□□□□□□□□□□□□□□□□CAPTCHA□□□□

□□□□□□□□□□□□□□□CAPTCHA□□□□—□□□□□□□□□□□□□□□CAPTCHA□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□get_captcha □□□□□□□□□□□□□□□□Image.new□□□□□□□□□□□□□□□□□self.image□□□

□□□□□□_add_noise_points □_add_noise_lines □□□

6.7.4 □□□□

□□□7□□□□□□□□□□□□□□□□□□□□□□□□□□helper□□

□□□□□□□□□□□□□□□CAPTCHA□□ get_english_captcha □□□□□□□□□

□ Unix □ □ □ □
□/usr/share/dict/words □□□□□□□□□□□□□□□□□□□□□□

```
def get_english_captcha(self):
    words = '/usr/share/dict/words'
    with open(words, 'r') as wf:
        words = wf.readlines()
        aword = random.choice(words)
        aword = aword.strip() # remove newline and spaces
    return self.get_captcha(text=aword)
```

□□□□□CAPTCHA□□□□□□□□□□□□□□□□□□□□□□□□
□□□

□□□□□□Web□□□□□□□□□□□□□□□□□□Python□□□□□
□□□□□□□Web□□□□□□

□ □ □ □ □ Web □ □ □ □ □ □ □ □ □ □ Python □ □ recaptcha-client
□https://pypi.python. org/pypi/recaptcha-client□□reCAPTCHA
□http://www.google.com/recaptcha□□□□□□□□□□□□□□□□
□□□□□□□reCAPTCHAWeb □□□□□□□□□□□□pycrypto□□□□
□□Web□□□□□□□□□□□□□□□OCR□□□Google□□□□□□□□□
□□□□□□□□□reCAPTCHA□□□□□□□□□

□□

[1]. EEG: electroencephalo-graph, □□□
[2]. Windows □□□ sudo □□□□□□□□□□□□□□□□□□□□□□
[3]. CAPTCHA□□□□□□□□□□□□□□□□□□□□

[illegible]7[illegible] [illegible]

[illegible]

◆ [illegible]

◆ [illegible]

◆ [illegible]

◆ [illegible]

◆ [illegible]

◆ [illegible]

◆ [illegible]

◆ [illegible]

7.1 [illegible]

[illegible]“[illegible]”[illegible]

7.2 [illegible]

□□

7.2.1 □□□□

□□□

□□□□□□□□□□□□□□□□□□□□□□

◆ □□□□□□□□□□□□□□□□□□

◆ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

◆ □□□□□□□□□□□□□□□□□□□□□□

□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□10□□□□□□□□□□□□□□2□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□

□□□

□□

7.2.2 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□

□□□□□□□□□□□

◆ □□□□□□□□/□□ y □□□ z□

◆ □□□□□□□□□

◆ □□□□□□ y □□□□□□□□□□□□

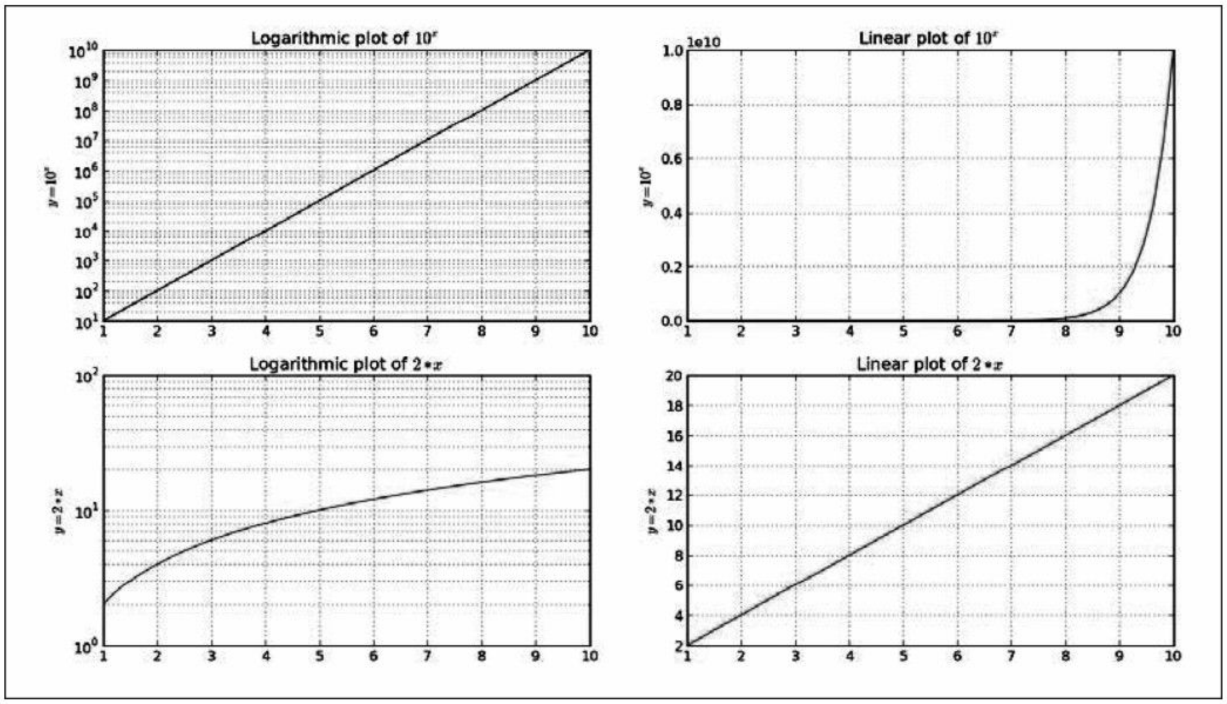
◆ □□□□□□ z □□□□□□□□□□□□

□□□

```
from matplotlib import pyplot as plt
import numpy as np
x = np.linspace(1, 10)
y = [10 ** el for el in x]
z = [2 * el for el in x]
fig = plt.figure(figsize=(10, 8))
ax1 = fig.add_subplot(2, 2, 1)
ax1.plot(x, y, color='blue')
ax1.set_yscale('log')
ax1.set_title(r'Logarithmic plot of $ {10}^{x} $ ')
ax1.set_ylabel(r'$ {y} = {10}^{x} $')
plt.grid(b=True, which='both', axis='both')
ax2 = fig.add_subplot(2, 2, 2)
ax2.plot(x, y, color='red')
ax2.set_yscale('linear')
ax2.set_title(r'Linear plot of $ {10}^{x} $ ')
ax2.set_ylabel(r'$ {y} = {10}^{x} $')
plt.grid(b=True, which='both', axis='both')
ax3 = fig.add_subplot(2, 2, 3)
```

```
ax3.plot(x, z, color='green')
ax3.set_yscale('log')
ax3.set_title(r'Logarithmic plot of $ {2}*{x} $ ')
ax3.set_ylabel(r'$ {y} = {2}*{x} $')
plt.grid(b=True, which='both', axis='both')
ax4 = fig.add_subplot(2, 2, 4)
ax4.plot(x, z, color='magenta')
ax4.set_yscale('linear')
ax4.set_title(r'Linear plot of $ {2}*{x} $ ')
ax4.set_ylabel(r'$ {y} = {2}*{x} $')
plt.grid(b=True, which='both', axis='both')
```

□□□□□□□7-1□□□□□□



□7-1

7.2.3 □□□□

[illegible]y[illegible]z[illegible]y[illegible]x[illegible]z[illegible]x[illegible]

[illegible]x[illegible]y[illegible]x[illegible]z[illegible]

[illegible]y [illegible]y [illegible]set_yscale('log')[illegible]

[illegible]

[illegible] plt.grid(b=True, which='both', axis='both')[illegible]

[illegible]

7.3 [illegible]

[illegible]

[illegible]

[illegible]x[illegible]y[illegible]—[illegible] 3D [illegible] z [illegible]3D [illegible]2D[illegible]

7.3.1 [illegible]

[illegible]Python[illegible]

[illegible] WAV [illegible]libsndfile1 [illegible]/[illegible]Ubuntu[illegible]

```
$ sudo apt-get install libasound1-dev
```

[illegible]dev[illegible]pip[illegible]scikits.audiolab[illegible]

[illegible] libasound [illegible] ALSA [illegible] Advanced Linux Sound Architecture[illegible]Linux[illegible]ALSA[illegible] Ubuntu Linux[illegible]

```
$ sudo apt-get install libasound2-dev
```

[illegible]pip[illegible]WAV[illegible]scikits.audiolab[illegible]

```
$ pip install scikits.audiolab
```

[illegible]

7.3.2 [illegible]

[illegible] test.wav[illegible]

[illegible]

1.[illegible]WAV[illegible]

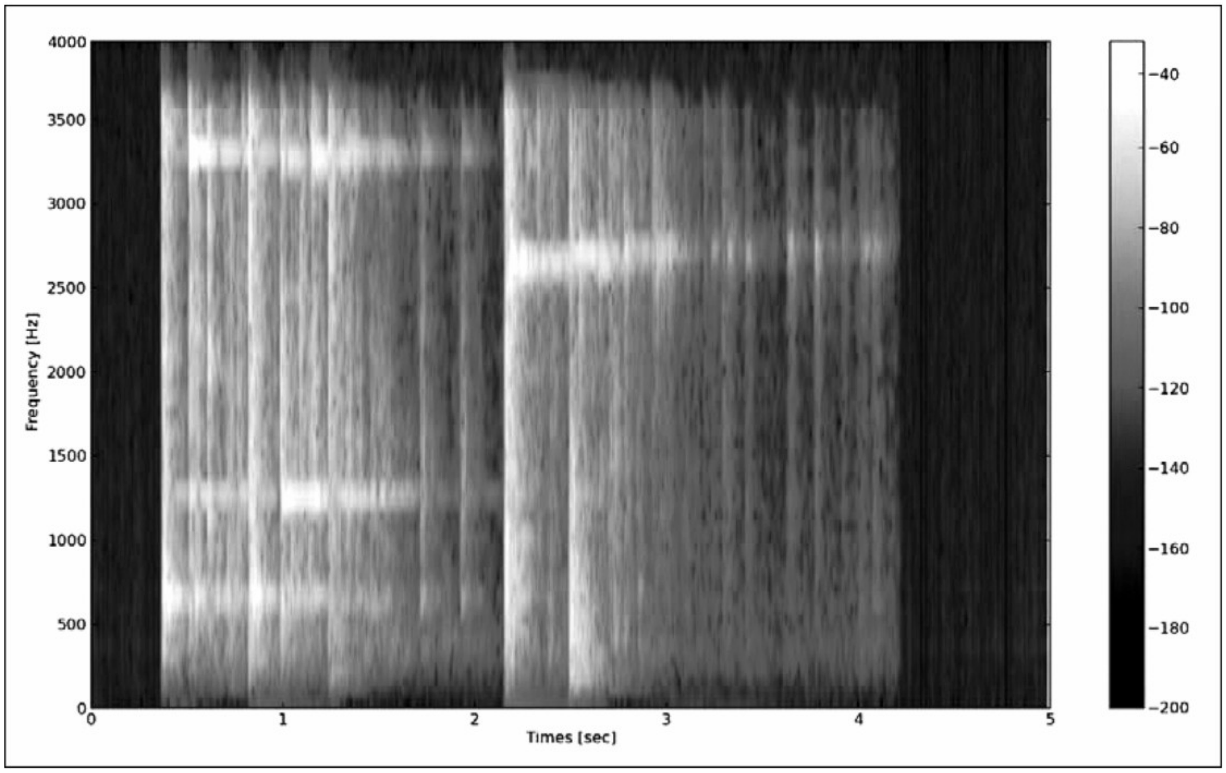
2.[illegible]NFFT[illegible]

3.[illegible]noverlap[illegible]

```
import os
from math import floor, log
from scikits.audiolab import Sndfile
import numpy as np
from matplotlib import pyplot as plt
# Load the sound file in Sndfile instance
soundfile = Sndfile("test.wav")
```

```
# define start/stop seconds and compute start/stop frames
start_sec = 0
stop_sec = 5
start_frame = start_sec * soundfile.samplerate
stop_frame = stop_sec * soundfile.samplerate
# go to the start frame of the sound object
soundfile.seek(start_frame)
# read number of frames from start to stop
delta_frames = stop_frame - start_frame
sample = soundfile.read_frames(delta_frames)
map = 'CMRmap'
fig = plt.figure(figsize=(10, 6), )
ax = fig.add_subplot(111)
# define number of data points for FT
NFFT = 128
# define number of data points to overlap for each block
noverlap = 65
pxx, freq, t, cax = ax.specgram(sample, Fs=soundfile.samplerate,
    NFFT=NFFT, noverlap=noverlap,
    cmap=plt.get_cmap(map))
plt.colorbar(cax)
plt.xlabel("Times [sec]")
plt.ylabel("Frequency [Hz]")
plt.show()
```

[illegible]7-2[illegible]



[illegible]7-2

NFFT[illegible]NFFT[illegible]2[illegible]noverlap[illegible]

7.3.3 [illegible]

[illegible] scikits.audiolab.SndFile [illegible]

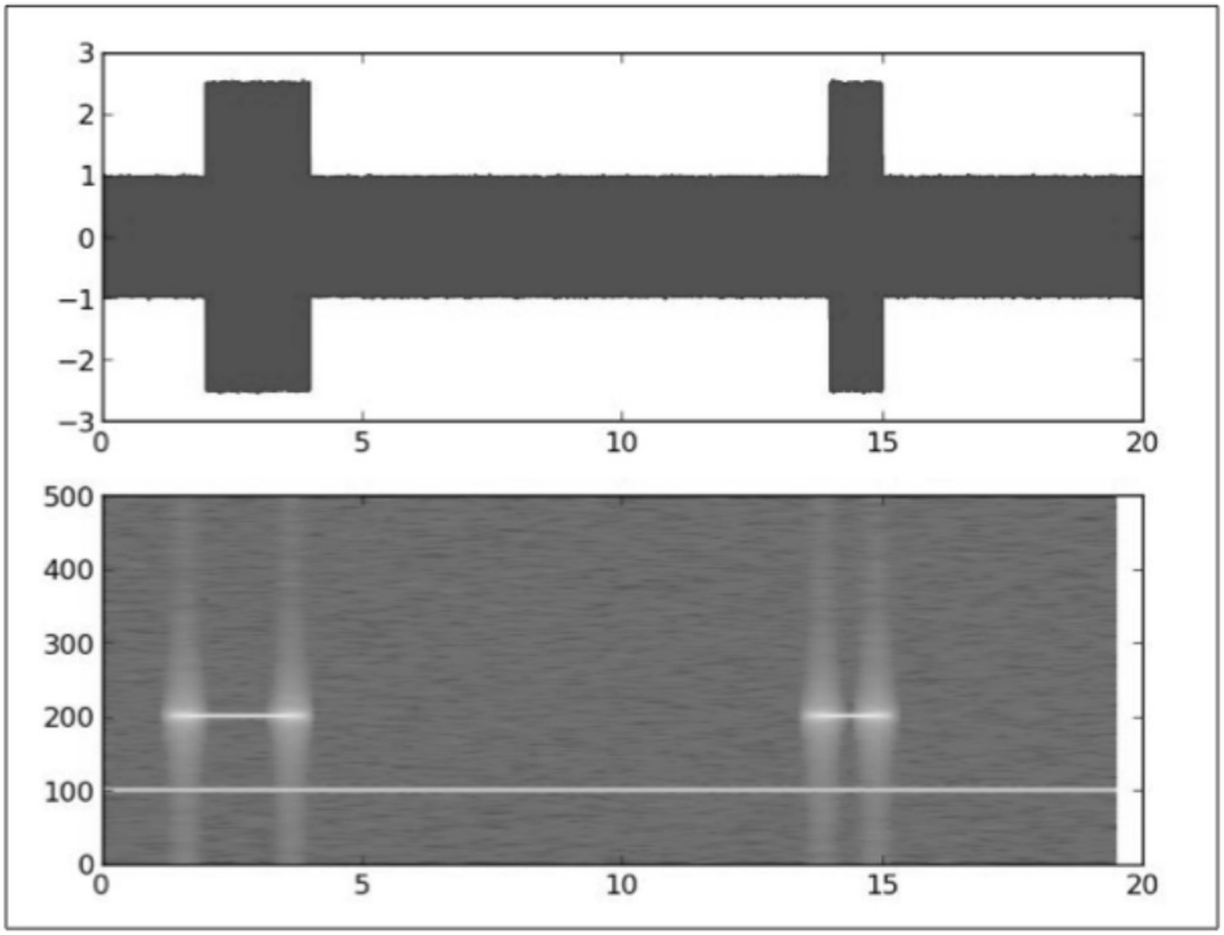
[illegible]read_frames()[illegible]start, end[illegible]

7.3.4 [illegible]

□□□□□□□□□□wave□□□□□□□□□□□□□□□□□□□□□

```
import numpy
def _get_mask(t, t1, t2, lvl_pos, lvl_neg):
    if t1 >= t2:
        raise ValueError("t1 must be less than t2")
    return numpy.where(numpy.logical_and(t > t1, t < t2), lvl_pos, lvl_neg)
def generate_signal(t):
    sin1 = numpy.sin(2 * numpy.pi * 100 * t)
    sin2 = 2 * numpy.sin(2 * numpy.pi * 200 * t)
    # add interval of high pitched signal
    sin2 = sin2 * get_mask□t,2,5,1.0,0.0□
    noise = 0.02 * numpy.random.randn(len(t))
    final_signal = sin1 + sin2 + noise
    return final_signal
if __name__ == '__main__':
    step = 0.001
    sampling_freq=1000
    t = numpy.arange(0.0, 20.0, step)
    y = generate_signal(t)
    # we can visualize this now
    # in time
    ax1 = plt.subplot(211)
    plt.plot(t, y)
    # and in frequency
    plt.subplot(212)
    plt.specgram(y, NFFT=1024, noverlap=900,
        Fs=sampling_freq, cmap=plt.cm.gist_heat)
    plt.show()
```

[illegible]7-3[illegible]x[illegible]y[illegible]x[illegible]y[illegible]



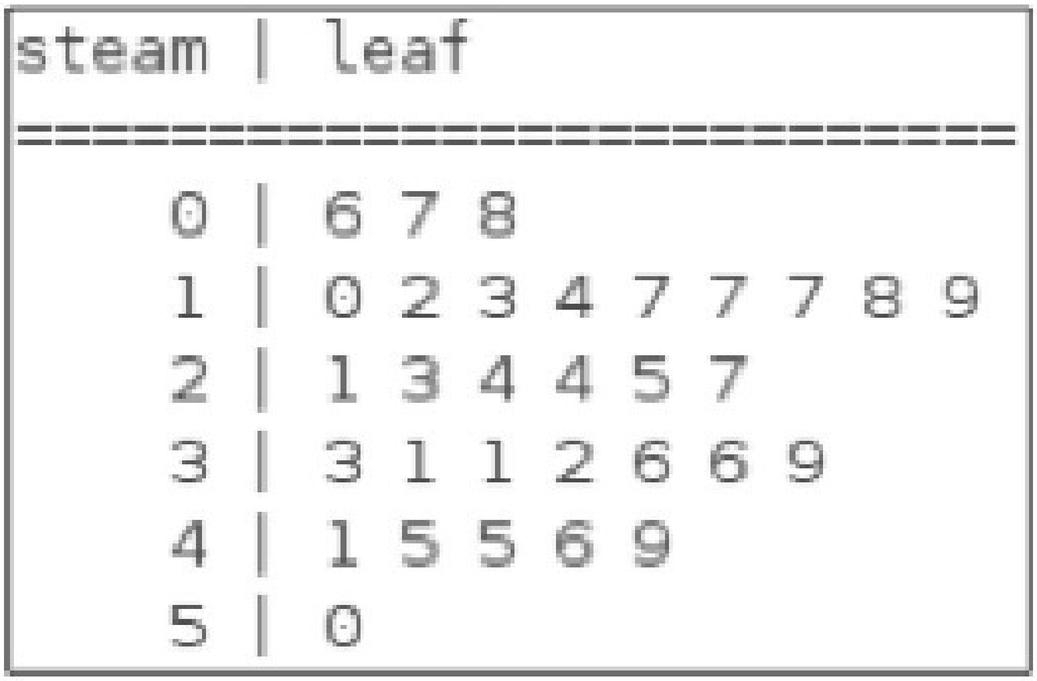
[illegible]7-3

7.4 [illegible]

[illegible]stem plot[illegible] x [illegible]y[illegible]

□□□□□□□□□□□□□□

□□□□□□□□□□□□□stem and leaf plot□□□7-4□□□



```
steam | leaf
==============================
    0 | 6 7 8
    1 | 0 2 3 4 7 7 7 8 9
    2 | 1 3 4 4 5 7
    3 | 3 1 1 2 6 6 9
    4 | 1 5 5 6 9
    5 | 0
```

□7-4

7.4.1 □□□□

□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□y=0□□□□□□□□□□□

7.4.2 □□□□

□□□□□matplotlib□stem()□□□□□□□□□□□□□□□□□□□□□□y□□x□□□□□□□□0□len(y)-1□□□□□□□□□x□y□□□□□□stem()□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

◆ linefmt[illegible]

◆ markerfmt[illegible]

◆ basefmt[illegible]

◆ label[illegible]

◆ hold[illegible]

◆ bottom[illegible] y [illegible] 0[illegible]

[illegible] hold [illegible]True[illegible]

[illegible]

[illegible]

1.[illegible]

2.[illegible]

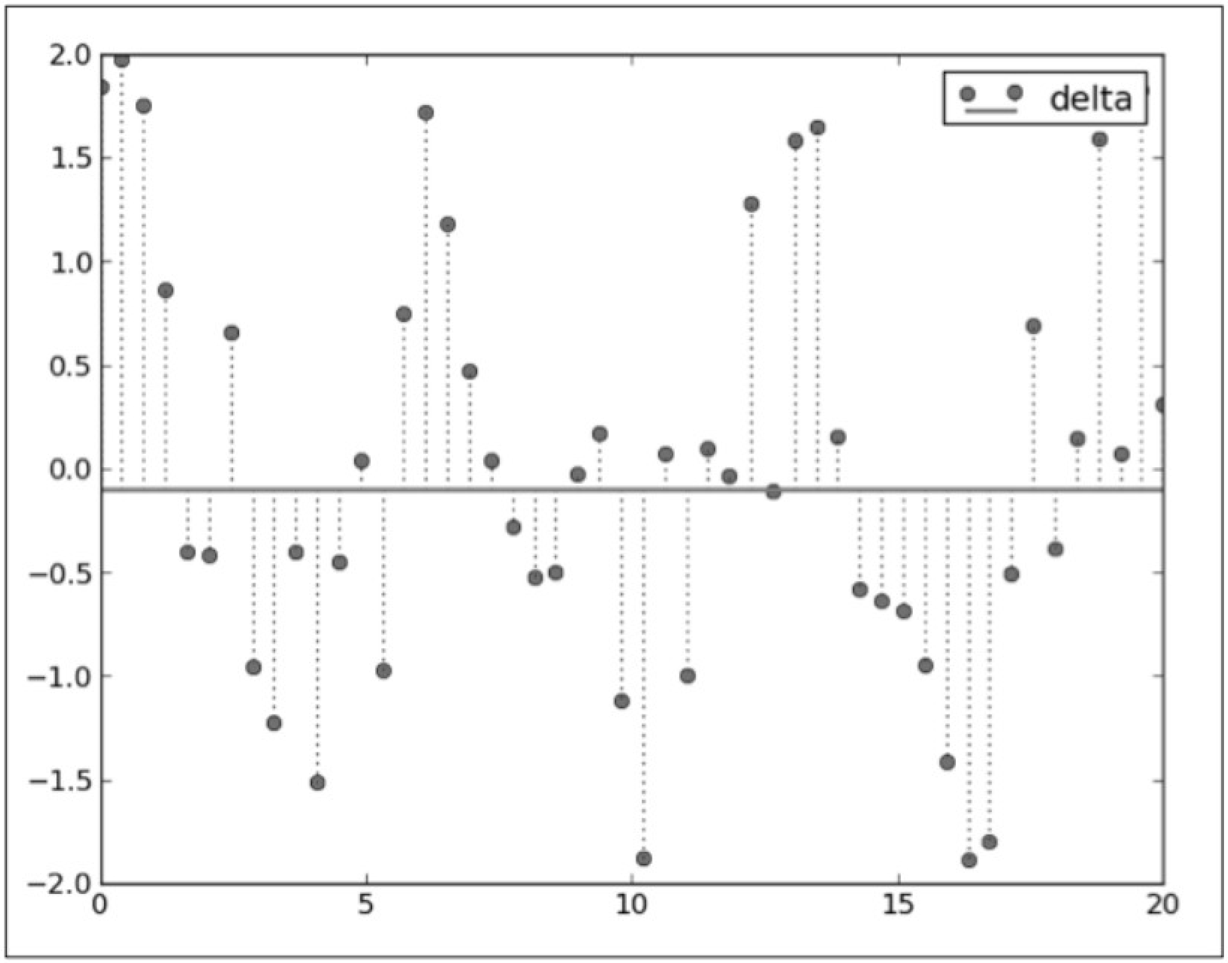
3.[illegible]

[illegible]

```
import matplotlib.pyplot as plt
import numpy as np
# time domain in which we sample
x = np.linspace(0, 20, 50)
# random function to simulate sampled signal
y = np.sin(x + 1) + np.cos(x ** 2)
# here we can setup baseline position
bottom = -0.1
# True -- hold current axes for further plotting
# False -- opposite. clear and use new figure/plot
hold = False
# set label for legend.
label = "delta"
```

```
markerline, stemlines, baseline = plt.stem(x, y, bottom=bottom,
    label=label, hold=hold)
# we use setp() here to setup
# multiple properties of lines generated by stem()
plt.setp(markerline, color='red', marker='o')
plt.setp(stemlines, color='blue', linestyle=':')
plt.setp(baseline, color='grey', linewidth=2, linestyle='-')
# draw a legend
plt.legend()
plt.show()
```

[illegible]7-5[illegible]



□7-5

7.4.3 □□□□

□□ Numpy □numpy.linspace□numpy.cos□numpy.sin□□□□□□

□□□□□□□□□□□□□□□□□□□□□0.0□

□□□□□□□□□□□□□hold□□True□□□□□□□□□□□□□□□□□□

□□□□matplotlib.stem□□□□□□□□□markerline□□□Line2D□□□

Line2D□□□□□□□□□□□□□□□□□□□□□□□□□□ baseline □□□□ Line2D □□□□□□□□ stemlines □□□□□□□□□□□□□□□□□□□□□ stemlines□□□□□□□□Line2D□□□□□□□Python□□□

□□setp□□□□□□□□□□□□□□□□□□□□□Line2D□□□□□□□□ □□□□□□□□□□□□□□□□

□□□□□□□□□□□setp□□□□□□□□□

7.5 □□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□ □□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

7.5.1 □□□□

□matplotlib□matplotlib.pyplot.streamplot□□□□□□□□□□□ □□□ □□□□□□□□□□□□□□□□□□□

□□□□□□□□□□X□Y□□□□ Numpy □□□□□□□□□U□V□□□□□□ □X□Y□□□□□Numpy□□□U□V□□□□□□□□□□□Y□□□□□□□□□□□X□ □□□

□□□□□□□□□□□□□□□ linewidth □□□□□□□□□□□ u□v□□□□ □□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□linewidth□□□□□□□□□□

□□□FancyArrowPatch □□□□□□□□□□□□□□□□□□□□□□□□ arrowsize□□□□□□□arrowstyle□□□□□□□□“simple”□“->”…□□

7.5.2 [illegible]

[illegible]

1.[illegible]

2.[illegible]

3.[illegible]

4.[illegible]

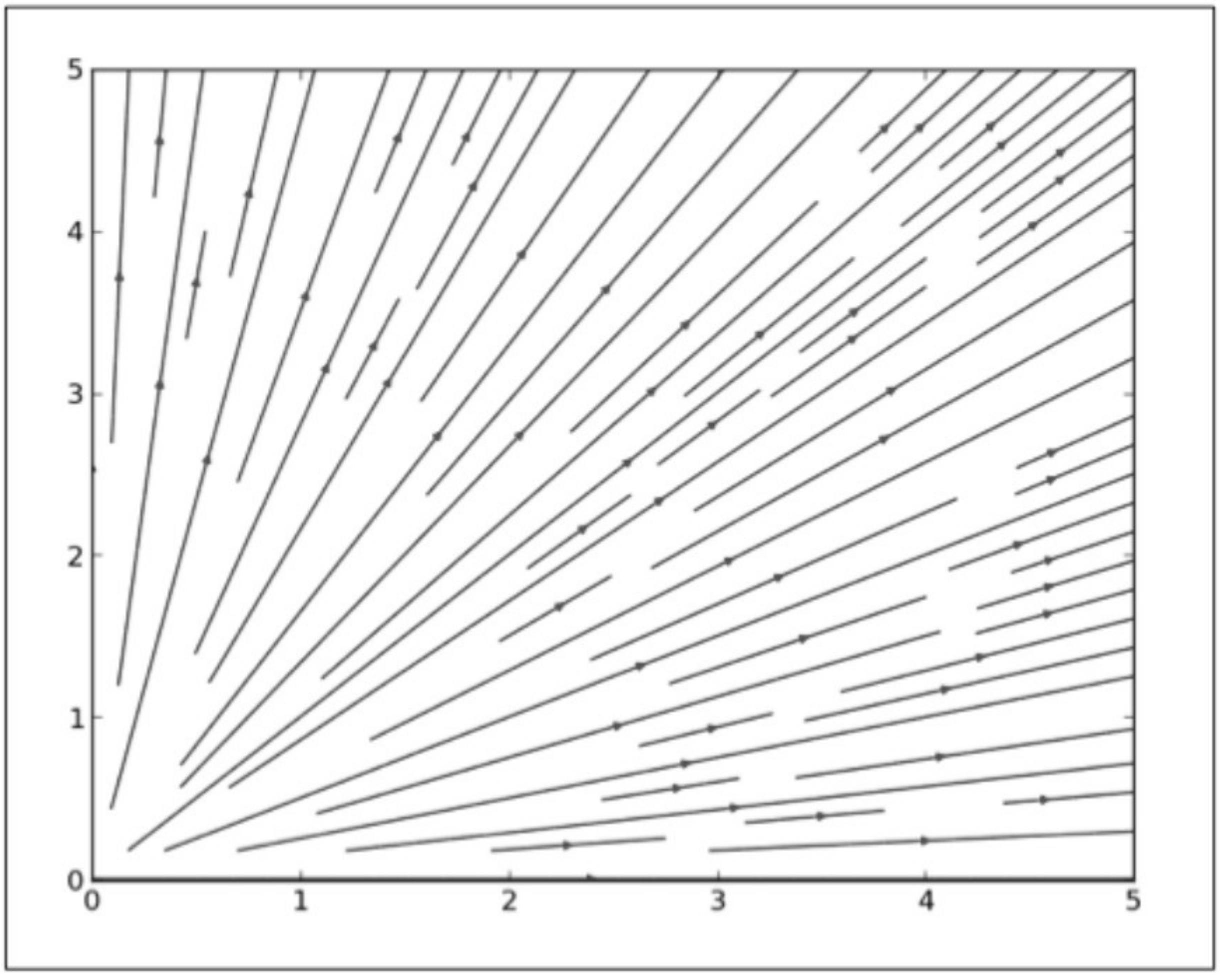
[illegible]

```
import matplotlib.pyplot as plt
import numpy as np
Y, X = np.mgrid[0:5:100j, 0:5:100j]
U = X
V = Y
from pprint import pprint
print "X"
pprint(X)
print "Y"
pprint(Y)
plt.streamplot(X, Y, U, V)
plt.show()
```

[illegible]

```
X
array([[   0.          ,      0.05050505,      0.1010101   ,   ...,
4.8989899 ,
    4.94949495,   5.        ],
    [ 0.        ,   0.05050505,   0.1010101 , ...,   4.8989899 ,
    4.94949495,   5.        ],
    [ 0.        ,  0.05050505,  0.1010101 , ...,  4.8989899 ,
    4.94949495,  5.        ],
    ...,
    [ 0.        ,  0.05050505,  0.1010101 , ...,  4.8989899 ,
    4.94949495,  5.        ],
    [ 0.        ,  0.05050505,  0.1010101 , ...,  4.8989899 ,
    4.94949495,  5.        ],
    [ 0.        ,  0.05050505,  0.1010101 , ...,  4.8989899 ,
    4.94949495,  5.        ]])
  Y
  array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
    0.        ,  0.        ],
    [  0.05050505,     0.05050505,     0.05050505,  ...,
0.05050505,
    0.05050505,  0.05050505],
    [  0.1010101  ,     0.1010101  ,     0.1010101  ,  ...,
0.1010101 ,
    0.1010101 ,  0.1010101 ],
    ...,
    [  4.8989899  ,     4.8989899  ,     4.8989899  ,  ...,
4.8989899 ,
    4.8989899 , 4.8989899 ],
    [     4.94949495,     4.94949495,     4.94949495,     ...,
4.94949495,
    4.94949495, 4.94949495],
    [ 5.        ,  5.        ,  5.        , ...,  5.        ,
    5.        ,  5.        ]])
```

□□□□□□□□□□7-6□□□



□7-6

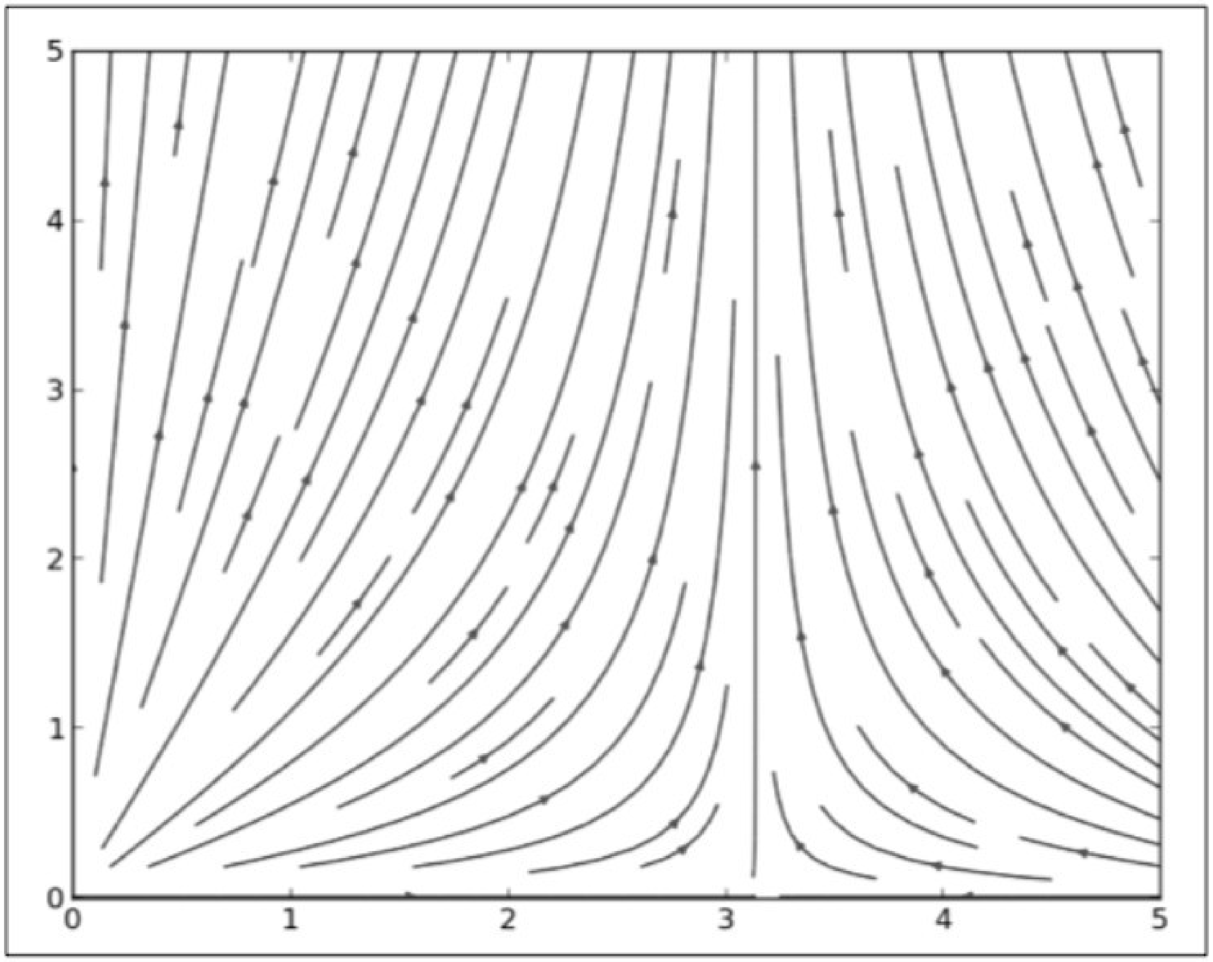
7.5.3 □□□□

□□Numpy□mgrid□□□□□□□□□□□□□□□□□□□□X□Y□□□□□□□□□□□□□□□□□□□□□□□□□-2□2□□

□□meshgrid□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□ U □ V □□□□□□□ U □ V □□□□□□□□□□□□□□□□□□ U =np.sin(X)□□ V = sin(Y)□□□□□□□□□□□□□□□□□□□□□ 7-7 □ U

=np.sin(X)[illegible]



[illegible]7-7

[illegible]matplotlib[illegible]

7.5.4 [illegible]

[illegible]matplotlib[illegible]

[illegible]

7.6 [illegible]

[illegible]

[illegible]

7.6.1 [illegible]

[illegible]/[illegible]

[illegible]/[illegible]

[illegible] matplotlib [illegible]

[illegible]

◆ Sequential[illegible]

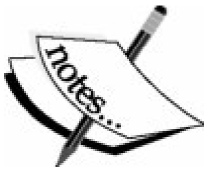
◆ Diverging[illegible]0[illegible]

◆ Qualitative[illegible]

◆ Cyclic[illegible]

matplotlib[illegible]autumn[illegible]bone[illegible]cool[illegible]copper[illegible]flag[illegible]gray[illegible] hot□hsv□jet□pink□prism□sprint□summer□winter□spectral□

□Yorick□□□□□□□□□□□□□□□□□□□□□□□GIST□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□gist_□□□



Yorick□□□□□□□□□□□□□□□C□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□http://yorick.sourceforge.net/index.php□□□□□□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ gist_earth □ gist_heat □ gist_ncar □ gist_rainbow□gist_stern□

□□□□□□ColorBrewer□http://colorbrewer.org□□□□□□□□□□□□□□□□□□□□

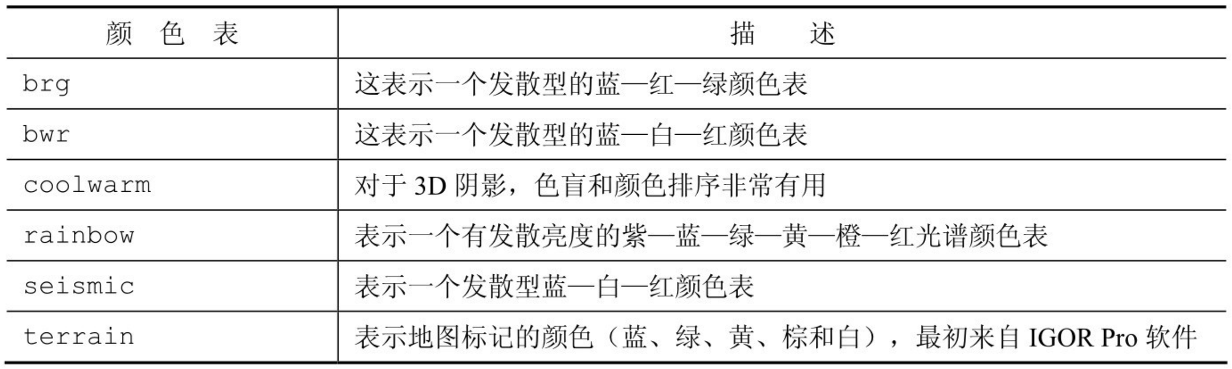
◆ Diverging□□□□□□□□□□□□□□□□

◆ Sequential□□□□□□□□□□

◆ Qualitative□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□7-1□□□

□7-1



颜　色　表	描　　述
brg	这表示一个发散型的蓝—红—绿颜色表
bwr	这表示一个发散型的蓝—白—红颜色表
coolwarm	对于 3D 阴影，色盲和颜色排序非常有用
rainbow	表示一个有发散亮度的紫—蓝—绿—黄—橙—红光谱颜色表
seismic	表示一个发散型蓝—白—红颜色表
terrain	表示地图标记的颜色（蓝、绿、黄、棕和白），最初来自 IGOR Pro 软件

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□_r □□□□□□□□□□hot_r□□□□□□□hot□□□□

7.6.2 □□□□

□ matplotlib □□□□□□□□□□□□□□□□□□ image, pcolor □ scatter □□□□□ cmap □□□□□□□□□□□□□□□□□□□□ colors.Colormap□□□

□□□□matplotlib.pyplot.set_cmap□□□□□□□□□□□□□□ cmap□

□□ matplotlib.pyplot.colormaps □□□□□□□□□□□□□□□ IPython□□□□□□

```
In [1]: import matplotlib.pyplot as plt
In [2]: plt.colormaps()
Out[2]:
 ['Accent',
 'Accent_r',
 'Blues',
 'Blues_r',
 ...
 'winter',
 'winter_r']
```

□□□□□□□□□□□□□□□□□140□□□□□□□□

□□□□□pyplot□□□□□□□colormaps□□colormaps□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

1.□□ColorBrewer□□□□□□□□□diverging□□□□□

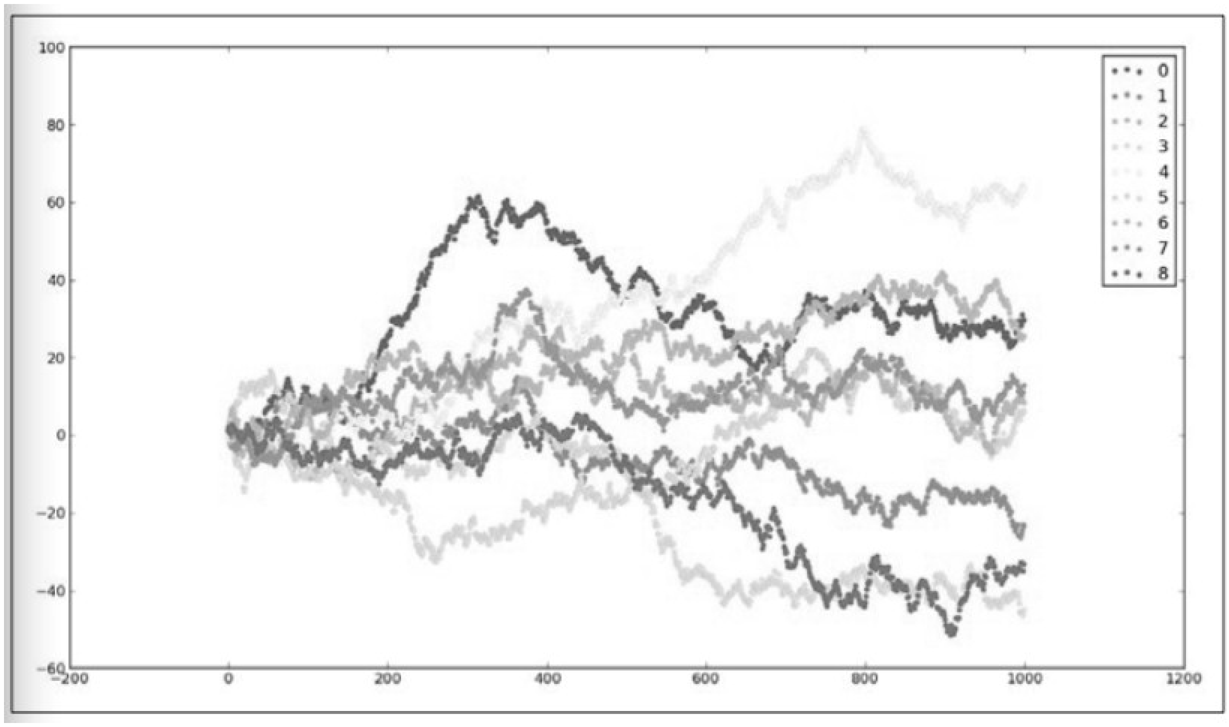
2.□□□□x□y□□□y□□□□□□□□□□□□□□

3.□matplotlib□□□□□□□□□□

4.□□□□□□□□□□□□□□□□□

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
# Red Yellow Green divergent colormap
red_yellow_green = ['#d73027', '#f46d43', '#fdae61',
   '#fee08b', '#ffffbf', '#d9ef8b',
   '#a6d96a', '#66bd63', '#1a9850']
sample_size = 1000
fig, ax = plt.subplots(1)
for i in range(9):
   y = np.random.normal(size=sample_size).cumsum()
   x = np.arange(sample_size)
   ax.scatter(x, y, label=str(i), linewidth=0.1,
   edgecolors='grey',
      facecolor=red_yellow_green[i])
ax.legend()
plt.show()
```

[illegible]7-8[illegible]



[illegible]7-8

7.6.3 [illegible]

[illegible]ColorBrewer[illegible]—[illegible]—[illegible]diverging[illegible]

ColorBrewer [illegible] Cynthia Brewer[illegible]Mark Harrower [illegible] Web [illegible] http://colorbrewer2.org/index.php?type=diverging& scheme=RdYlGn&n=9[illegible]

[illegible] matplotlib.rcParams [illegible]

□□□□□□ matplotlib □□□□□□□□□□□□□ matplotlib. rcParams['axes.cycle_color']□

7.6.4 □□□□

□□ matplotlib.pyplot.register_cmap □□□□□□□□□□□ matplotlib□□□□□get_cmap□□□□□□□□□□□□□□□□□□ □□□□

◆ register_cmap(name='swirly', cmap=swirly_cmap)

◆ register_cmap(name='choppy', data=choppydata, lut=128)

□□□□□□□□□□colors.Colormap□□□□□name□□□□□ name□□□□□□□□□□□cmap□□□name□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ matplotlib□

□□□□□name□□□matplotlib.pyplot.get_cmap□□□□□□ colors.Colormap□□

□ □ □ □ □ □ □ □ □ □ □ □ matplotlib.colors.LinearSegmented Colormap□□□□□□□

```
from pylab import *
cdict = {'red': ((0.0, 0.0, 0.0),
   (0.5, 1.0, 0.7),
   (1.0, 1.0, 1.0)),
   'green': ((0.0, 0.0, 0.0),
      (0.5, 1.0, 0.0),
      (1.0, 1.0, 1.0)),
   'blue': ((0.0, 0.0, 0.0),
          (0.5, 1.0, 0.0),
          (1.0, 0.5, 1.0))}
my_cmap = matplotlib.colors.LinearSegmentedColormap('my_colormap',cdict,256)
pcolor(rand(10,10),cmap=my_cmap)
colorbar()
```

[illegible]

[illegible]pylab[illegible]

```
imshow(X)
hot()
```

[illegible] X [illegible] cmap = 'hot'[illegible]

7.7 [illegible]

[illegible]

[illegible]

7.7.1 [illegible]

[illegible]

[illegible] X [illegible]Y[illegible]X[illegible]Y[illegible]

[illegible]

[illegible]1[illegible]-1[illegible]-0.5[illegible]0.5[illegible]

[illegible]B[illegible]A[illegible]A[illegible]B[illegible]

[illegible]

[illegible]

7.7.2 [illegible]

[illegible]

[illegible] Google Trends [illegible]CSV[illegible]

[illegible] ch07_search_data.py Python[illegible]

```
    # ch07_search_data
    # daily search trend for keyword 'flowers' for a year
    DATA = [
        1.04,  1.04,  1.16,  1.22,  1.46,  2.34,  1.16,  1.12,  1.24,
  1.30, 1.44,
        1.22, 1.26,
        1.34,  1.26,  1.40,  1.52,  2.56,  1.36,  1.30,  1.20,  1.12,
  1.12, 1.12,
```

1.06, 1.06,

1.00, 1.02, 1.04, 1.02, 1.06, 1.02, 1.04, 0.98, 0.98,
0.98, 1.00,

1.02, 1.02,

1.00, 1.02, 0.96, 0.94, 0.94, 0.94, 0.96, 0.86, 0.92,
0.98, 1.08,

1.04, 0.74,

0.98, 1.02, 1.02, 1.12, 1.34, 2.02, 1.68, 1.12, 1.38,
1.14, 1.16,

1.22, 1.10,

1.14, 1.16, 1.28, 1.44, 2.58, 1.30, 1.20, 1.16, 1.06,
1.06, 1.08,

1.00, 1.00,

0.92, 1.00, 1.02, 1.00, 1.06, 1.10, 1.14, 1.08, 1.00,
1.04, 1.10,

1.06, 1.06,

1.06, 1.02, 1.04, 0.96, 0.96, 0.96, 0.92, 0.84, 0.88,
0.90, 1.00,

1.08, 0.80,

0.90, 0.98, 1.00, 1.10, 1.24, 1.66, 1.94, 1.02, 1.06,
1.08, 1.10,

1.30, 1.10,

1.12, 1.20, 1.16, 1.26, 1.42, 2.18, 1.26, 1.06, 1.00,
1.04, 1.00,

0.98, 0.94,

0.88, 0.98, 0.96, 0.92, 0.94, 0.96, 0.96, 0.94, 0.90,
0.92, 0.96,

0.96, 0.96,

0.98, 0.90, 0.90, 0.88, 0.88, 0.88, 0.90, 0.78, 0.84, 0.86, 0.92,

1.00, 0.68,

0.82, 0.90, 0.88, 0.98, 1.08, 1.36, 2.04, 0.98, 0.96, 1.02, 1.20,

0.98, 1.00,

1.08, 0.98, 1.02, 1.14, 1.28, 2.04, 1.16, 1.04, 0.96, 0.98, 0.92,

0.86, 0.88,

0.82, 0.92, 0.90, 0.86, 0.84, 0.86, 0.90, 0.84, 0.82, 0.82, 0.86,

0.86, 0.84,

0.84, 0.82, 0.80, 0.78, 0.78, 0.76, 0.74, 0.68, 0.74, 0.80, 0.80,

0.90, 0.60,

0.72, 0.80, 0.82, 0.86, 0.94, 1.24, 1.92, 0.92, 1.12, 0.90, 0.90,

0.94, 0.90,

0.90, 0.94, 0.98, 1.08, 1.24, 2.04, 1.04, 0.94, 0.86, 0.86, 0.86,

0.82, 0.84,

0.76, 0.80, 0.80, 0.80, 0.78, 0.80, 0.82, 0.76, 0.76, 0.76, 0.76,

0.78, 0.78,

0.76, 0.76, 0.72, 0.74, 0.70, 0.68, 0.72, 0.70, 0.64, 0.70, 0.72,

0.74, 0.64,

0.62, 0.74, 0.80, 0.82, 0.88, 1.02, 1.66, 0.94, 0.94, 0.96, 1.00,

1.16, 1.02,

1.04, 1.06, 1.02, 1.10, 1.22, 1.94, 1.18, 1.12, 1.06, 1.06, 1.04,

1.02, 0.94,

0.94, 0.98, 0.96, 0.96, 0.98, 1.00, 0.96, 0.92, 0.90, 0.86, 0.82,

0.90, 0.84,

0.84, 0.82, 0.80, 0.80, 0.76, 0.80, 0.82, 0.80, 0.72, 0.72, 0.76,

0.80, 0.76,

0.70, 0.74, 0.82, 0.84, 0.88, 0.98, 1.44, 0.96, 0.88, 0.92, 1.08,

0.90, 0.92,

0.96, 0.94, 1.04, 1.08, 1.14, 1.66, 1.08, 0.96, 0.90, 0.86, 0.84,

0.86, 0.82,

0.84, 0.82, 0.84, 0.84, 0.84, 0.84, 0.82, 0.86, 0.82, 0.82, 0.86,

0.90, 0.84,

0.82, 0.78, 0.80, 0.78, 0.74, 0.78, 0.76, 0.76, 0.70, 0.72, 0.76,

0.72, 0.70,

0.64]

□□□□□□□□□□□□

1.[illegible] flowers [illegible] Google Trend [illegible]d[illegible]

2.[illegible]365 [illegible] Google Trend [illegible]d1[illegible]

3.[illegible]4[illegible]

4.[illegible]d[illegible]d1[illegible]

5.[illegible]d1[illegible]d1[illegible]

6.[illegible]d1[illegible]d1[illegible]

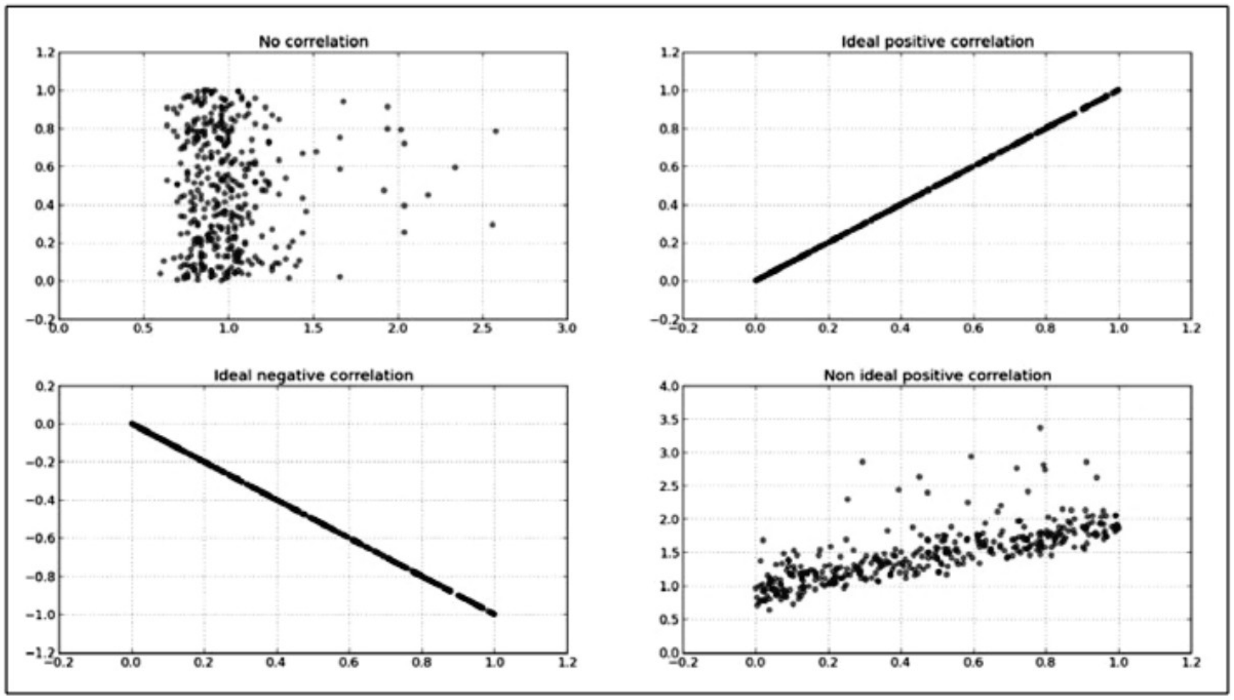
7.[illegible]d1[illegible]d1[illegible]d[illegible]

[illegible]

```
import matplotlib.pyplot as plt
import numpy as np
# import the data
from ch07_search_data import DATA
d = DATA
# Now let's generate random data for the same period
d1 = np.random.random(365)
assert len(d) == len(d1)
fig = plt.figure()
ax1 = fig.add_subplot(221)
ax1.scatter(d, d1, alpha=0.5)
ax1.set_title('No correlation')
ax1.grid(True)
ax2 = fig.add_subplot(222)
ax2.scatter(d1, d1, alpha=0.5)
ax2.set_title('Ideal positive correlation')
ax2.grid(True)
```

```
ax3 = fig.add_subplot(223)
ax3.scatter(d1, d1*-1, alpha=0.5)
ax3.set_title('Ideal negative correlation')
ax3.grid(True)
ax4 = fig.add_subplot(224)
ax4.scatter(d1, d1+d, alpha=0.5)
ax4.set_title('Non ideal positive correlation')
ax4.grid(True)
plt.tight_layout()
plt.show()
```

□□□□□□□□□□□□□□7-9□□□□



□7-9

7.7.3 □□□□

[illegible]d1[illegible]d1[illegible]d1[illegible]d[illegible] d[illegible] d1 [illegible]d[illegible]

7.7.4 [illegible]

[illegible]x[illegible]y[illegible]

[illegible]—[illegible]scatterhist()[illegible]bin[illegible]

[illegible]

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
```

[illegible]x[illegible]y[illegible]figsize[illegible]

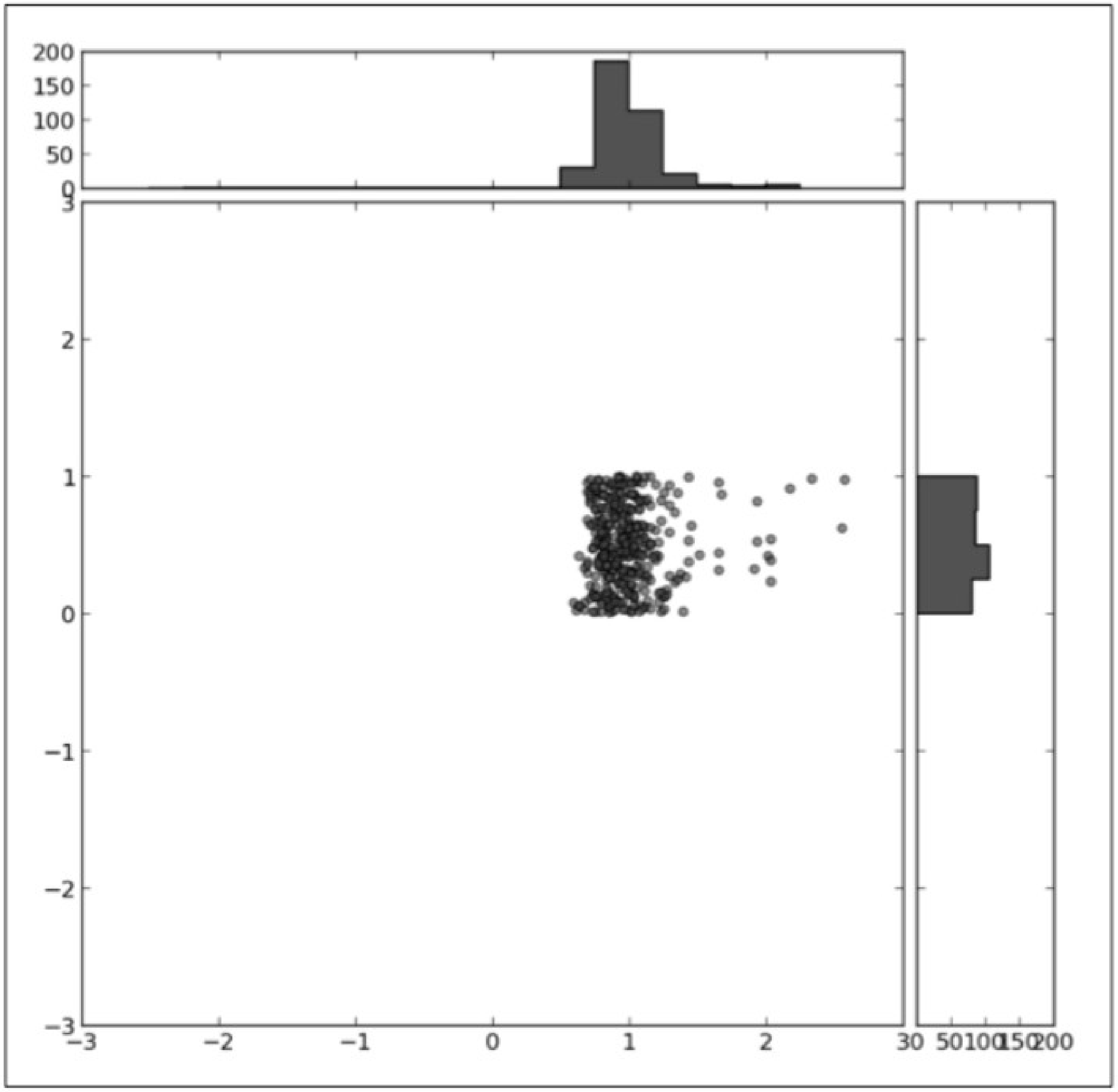
```
def scatterhist(x, y, figsize=(8,8)):
    """
    Create simple scatter & histograms of data x, y inside given plot
    @param figsize: Figure size to create figure
    @type figsize: Tuple of two floats representing size in inches
    @param x: X axis data set
```

```
        @type x: np.array
        @param y: Y axis data set
        @type y: np.array
        """
    _, scatter_axes = plt.subplots(figsize=figsize)
        # the scatter plot:
        scatter_axes.scatter(x, y, alpha=0.5)
        scatter_axes.set_aspect(1.)
        divider = make_axes_locatable(scatter_axes)
        axes_hist_x     =     divider.append_axes(position="top",
sharex=scatter_
    axes, size=1, pad=0.1)
        axes_hist_y = divider.append_axes(position="right",
    sharey=scatter_axes,
            size=1, pad=0.1)
        # compute bins accordingly
        binwidth = 0.25
        # global max value in both data sets
        xymax              =              np.max([np.max(np.fabs(x)),
np.max(np.fabs(y))])
        # number of bins
        bincap = int(xymax / binwidth) * binwidth
        bins = np.arange(-bincap, bincap, binwidth)
        nx, binsx, _ = axes_hist_x.hist(x, bins=bins,
    histtype='stepfilled',
            orientation='vertical')
        ny, binsy, _ = axes_hist_y.hist(y, bins=bins,
    histtype='stepfilled',
            orientation='horizontal')
        tickstep = 50
        ticksmax = np.max([np.max(nx), np.max(ny)])
        xyticks = np.arange(0, ticksmax + tickstep, tickstep)
        # hide x and y ticklabels on histograms
        for tl in axes_hist_x.get_xticklabels():
            tl.set_visible(False)
        axes_hist_x.set_yticks(xyticks)
        for tl in axes_hist_y.get_yticklabels():
            tl.set_visible(False)
        axes_hist_y.set_xticks(xyticks)
        plt.show()
```

[illegible]

```
if __name__ == '__main__': # import the data
    from ch07_search_data import DATA as d
    # Now let's generate random data for the same period
    d1 = np.random.random(365)
    assert len(d) == len(d1)
    # try with the random data
    # d = np.random.randn(1000)
    # d1 = np.random.randn(1000)
    scatterhist(d, d1)
```

[illegible]7-10[illegible]



[illegible]7-10

7.8 [illegible]

[illegible]

7.8.1 [illegible]

[illegible] pyplot lab [illegible] matplotlib [illegible] matplotlib.pyplot.xcorr [illegible] x[illegible]y[illegible]

[illegible]normed[illegible]True[illegible]0th[illegible]

[illegible]Numpy[illegible]numpy.correlate[illegible]

[illegible]usevlines[illegible]True[illegible]matplotlib[illegible]vlines()[illegible]plot()[illegible]plot()[illegible]Line2D[illegible]**kwargs[illegible]matplotlib.pyplot.xcorr[illegible]

7.8.2 [illegible]

[illegible]

1.[illegible]matplotlib.pyplot[illegible]

2.[illegible]numpy[illegible]

3.[illegible]Google[illegible]flowers[illegible]

4.[illegible]

5.[illegible]

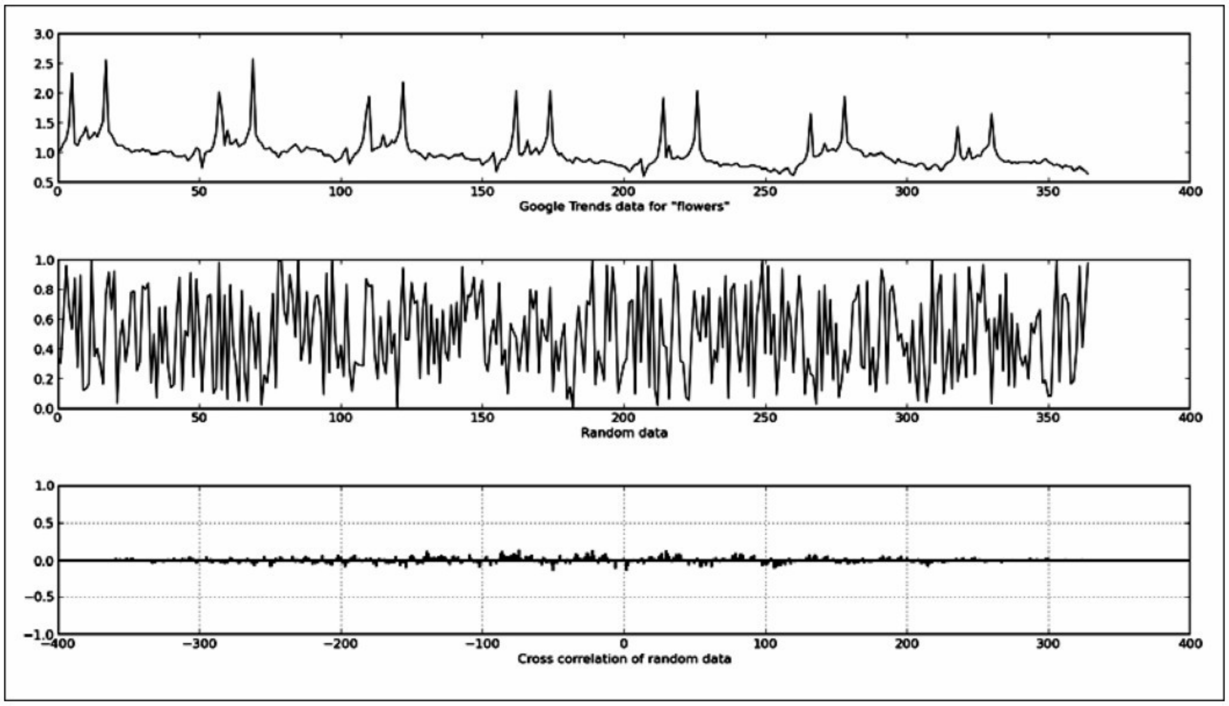
6.[illegible]

[illegible]

```
import matplotlib.pyplot as plt
import numpy as np
# import the data
from ch07_search_data import DATA as d
total = sum(d)
av = total / len(d)
```

```
z = [i - av for i in d]
# Now let's generate random data for the same period
d1 = np.random.random(365)
assert len(d) == len(d1)
total1 = sum(d1)
av1 = total1 / len(d1)
z1 = [i - av1 for i in d1]
fig = plt.figure()
# Search trend volume
ax1 = fig.add_subplot(311)
ax1.plot(d)
ax1.set_xlabel('Google Trends data for "flowers"')
# Random: "search trend volume"
ax2 = fig.add_subplot(312)
ax2.plot(d1)
ax2.set_xlabel('Random data')
# Is there a pattern in search trend for this keyword?
ax3 = fig.add_subplot(313)
ax3.set_xlabel('Cross correlation of random data')
ax3.xcorr(z, z1, usevlines=True, maxlags=None,
normed=True, lw=2)
ax3.grid(True)
plt.ylim(-1,1)
plt.tight_layout()
plt.show()
```

□□□□□□□□□7-11□□□□□□



□7-11

7.8.3 □□□□

□□ Google Trends □□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□matplotlib□xcorr□□□□□□□NumPy□correlate()□□□□□□□□□□□□□□□□□□□□□□

NumPy□□

□□n□□□□□□□□0.5□□□□

□□□□□□□□□□□□□□100s□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□100s□□□□□□□□□□□□□□□□□□□□□□□□□x=100□□□□□□□□□□□□□□□□

[illegible]

7.9 [illegible]

[illegible]

[illegible]

[illegible]

7.9.1 [illegible]

[illegible] matplotlib [illegible]365 [illegible] Google[illegible]365[illegible]

[illegible]

7.9.2 [illegible]

[illegible]

1.[illegible]matplotlib.pyplot[illegible]

2.[illegible]numpy[illegible]

3.[illegible]Google[illegible]

4.[illegible]

5.[illegible]NumPy[illegible]

6.[illegible]

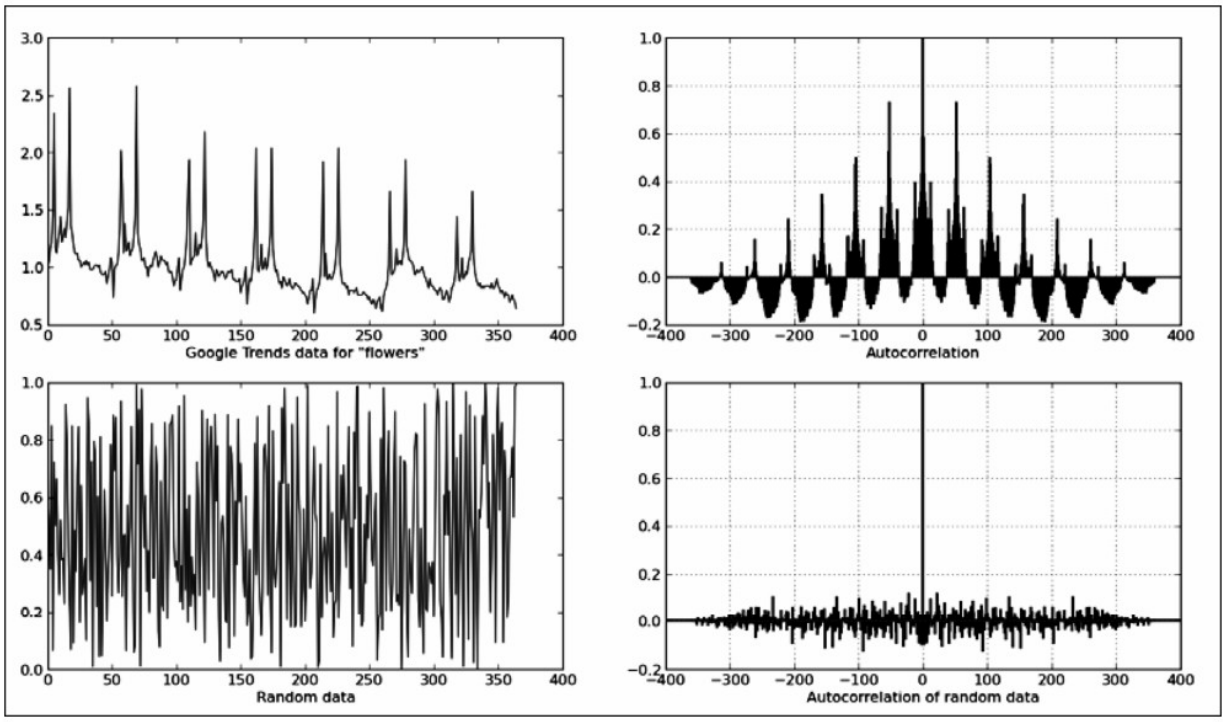
7.[illegible]

[illegible]

```
import matplotlib.pyplot as plt
import numpy as np
# import the data
from ch07_search_data import DATA as d
total = sum(d)
av = total / len(d)
z = [i - av for i in d]
fig = plt.figure()
# plt.title('Comparing autocorrelations')
# Search trend volume
ax1 = fig.add_subplot(221)
ax1.plot(d)
ax1.set_xlabel('Google Trends data for "flowers"')
# Is there a pattern in search trend for this keyword?
ax2 = fig.add_subplot(222)
ax2.acorr(z, usevlines=True, maxlags=None, normed=True, lw=2)
ax2.grid(True)
ax2.set_xlabel('Autocorrelation')
# Now let's generate random data for the same period
d1 = np.random.random(365)
assert len(d) == len(d1)
total = sum(d1)
av = total / len(d1)
```

```
    z = [i - av for i in d1]
    # Random: "search trend volume"
    ax3 = fig.add_subplot(223)
    ax3.plot(d1)
    ax3.set_xlabel('Random data')
    # Is there a pattern in search trend for this keyword?
    ax4 = fig.add_subplot(224)
    ax4.set_xlabel('Autocorrelation of random data')
    ax4.acorr( z, usevlines=True, maxlags=None,
normed=True, lw=2)
    ax4.grid(True)
    plt.show()
```

□□□□□□□□□7-12□□□□□□



□7-12

7.9.3 □□□□

[illegible]

[illegible]0[illegible] 0[illegible]

[illegible]——Google[illegible]0s[illegible]0s[illegible]30[illegible]60[illegible]110[illegible]Google[illegible]

[illegible]

7.9.4 [illegible]

[illegible] Python [illegible]Ljung-Box [illegible]Box-Pierce[illegible]

□8□ □□□matplotlib□□

□□□□□□□□□□

◆ □□□□□barbs□

◆ □□□□□

◆ □□□□□

◆ □□□□□

◆ □□□□□□□□□

◆ □ LaTeX □□□□

◆ □□ pyplot □ OO API □□□

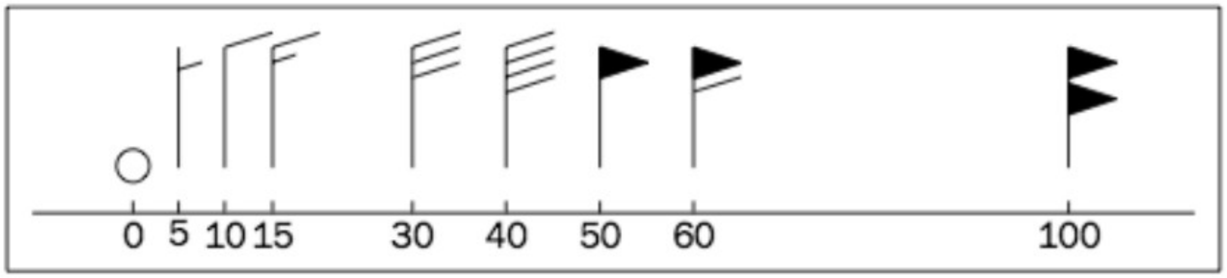
8.1 □□

□□□□□□□matplotlib□□□□□□□□□□□□□□□□□□□□matplotlib□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□matplotlib□□□□□□□□□□□□□□□□

8.2 □□□□□barbs□

□□□

[illegible]Python[illegible]matplotlib[illegible]8-1[illegible]



[illegible]8-1

[illegible]

[illegible]5[illegible]10[illegible]65[illegible]knots[illegible]

[illegible]0[illegible]5[illegible]10[illegible]15[illegible]30[illegible]40[illegible]40[illegible]50[illegible]60[illegible]100[illegible]0[illegible]

8.2.1 [illegible]

[illegible]matplotlib[illegible]matplotlib.pyplot.barbs[illegible]

barbs[illegible]X[illegible]Y[illegible]U[illegible]V[illegible]—[illegible]—[illegible]knots[illegible]

[illegible]

[illegible]pivot[illegible]

[illegible]

◆ barbcolor[illegible]

◆ flagcolor[illegible]

◆ facecolor[illegible] rcParams [illegible]

□□□□□□□□□□□□□facecolor□□□□□facecolor□□□□□□
□□□

□□□□sizes□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□

◆ spacing□□□□/□□□□□□□

◆ height□□□□□□□□□□□□□

◆ width□□□□□□□

◆ emptybarb□□□□□□□□□□□

8.2.2 □□□□

□□□□□□□□□□□□□□barb□□

1.□□□□□□□□□□□

2.□□□□□□□

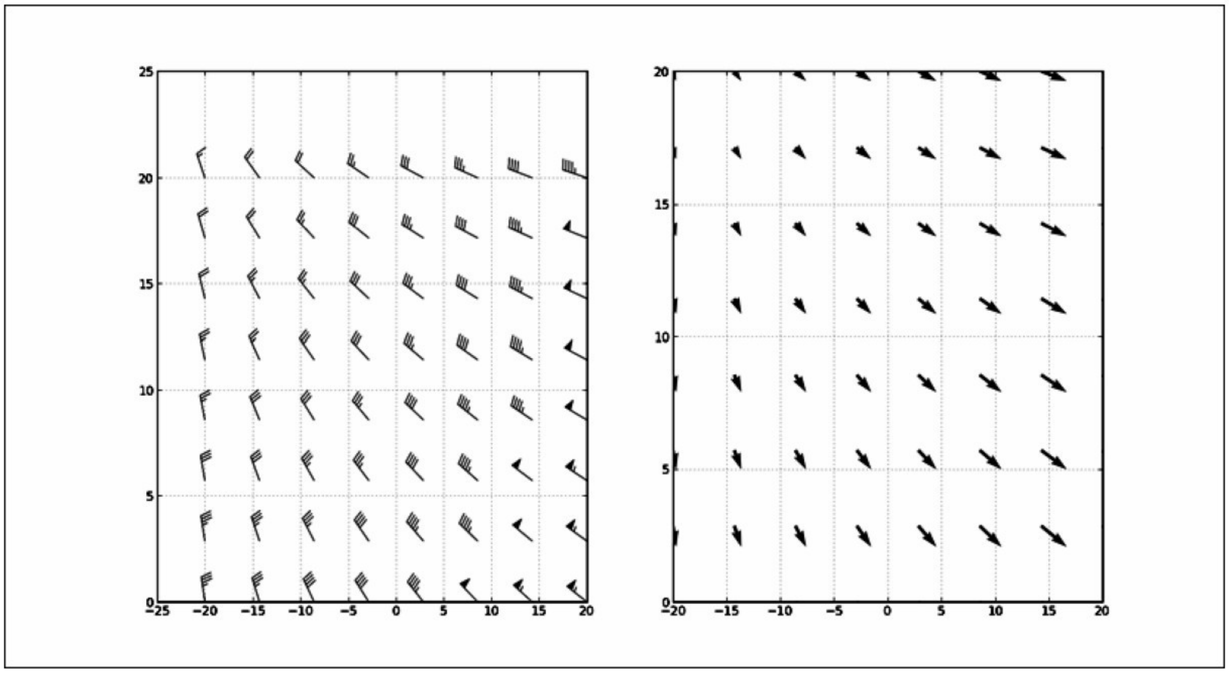
3.□□□□□

4.□□□□□□□□□□

□□□□□□□□

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-20, 20, 8)
y = np.linspace( 0, 20, 8)
# make 2D coordinates
X, Y = np.meshgrid(x, y)
U, V = X+25, Y-35
# plot the barbs
plt.subplot(1,2,1)
plt.barbs(X, Y, U, V, flagcolor='green', alpha=0.75)
plt.grid(True, color='gray')
# compare that with quiver / arrows
plt.subplot(1,2,2)
plt.quiver(X, Y, U, V, facecolor='red', alpha=0.75)
# misc settings
plt.grid(True, color='grey')
plt.show()
```

□□□□□□□□8-2□□□□□□□□



□8-2

8.2.3 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□matplotlib□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□NumPy□□□□□x□y□□□□□□□□□□□NumPy□meshgrid()□□□□□□□2D□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□U□V□□knots□□□□NS□□—□□□EW□□—□□□□□□□□□□□□□□□□□□□□□□□□□□□□□X□Y□□□□□□□□□

[illegible]

8.2.4 [illegible]

[illegible]

[illegible]barbs[illegible]flip_barb[illegible]True[illegible]False[illegible]

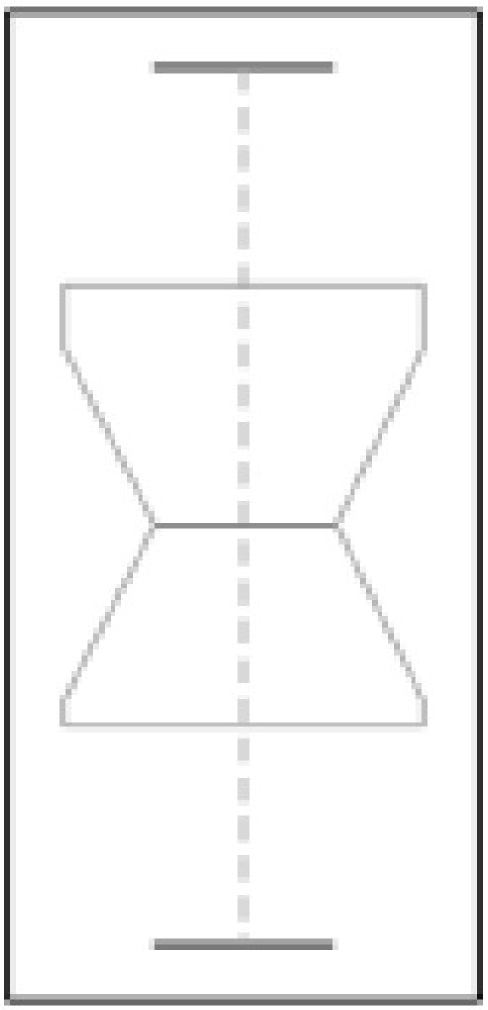
8.3 [illegible]

[illegible]

[illegible]

8.3.1 [illegible]

[illegible] 8-3 [illegible]



□8-3

□□□□□□□□□□□25%□□□□□□□□75%□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□ 1.5 □□□□□□□□□□□□□□□□□□□
99.3%□

□□□□□□□□□□□□□□□□□□[1]□□□□□□□□□□□□□

□□□

8.3.2 □□□□

□□□□□□□□□□□□□□matplotlib□□□□□□□□□□□□□

1.[illegible]

2.[illegible]PROCESSES[illegible]DATA[illegible]

3.[illegible]PROCESSES[illegible]LABELS[illegible]

4.[illegible]matplotlib.pyplot.boxplot[illegible]

5.[illegible]chartjunk[illegible][2][illegible]

6.[illegible]

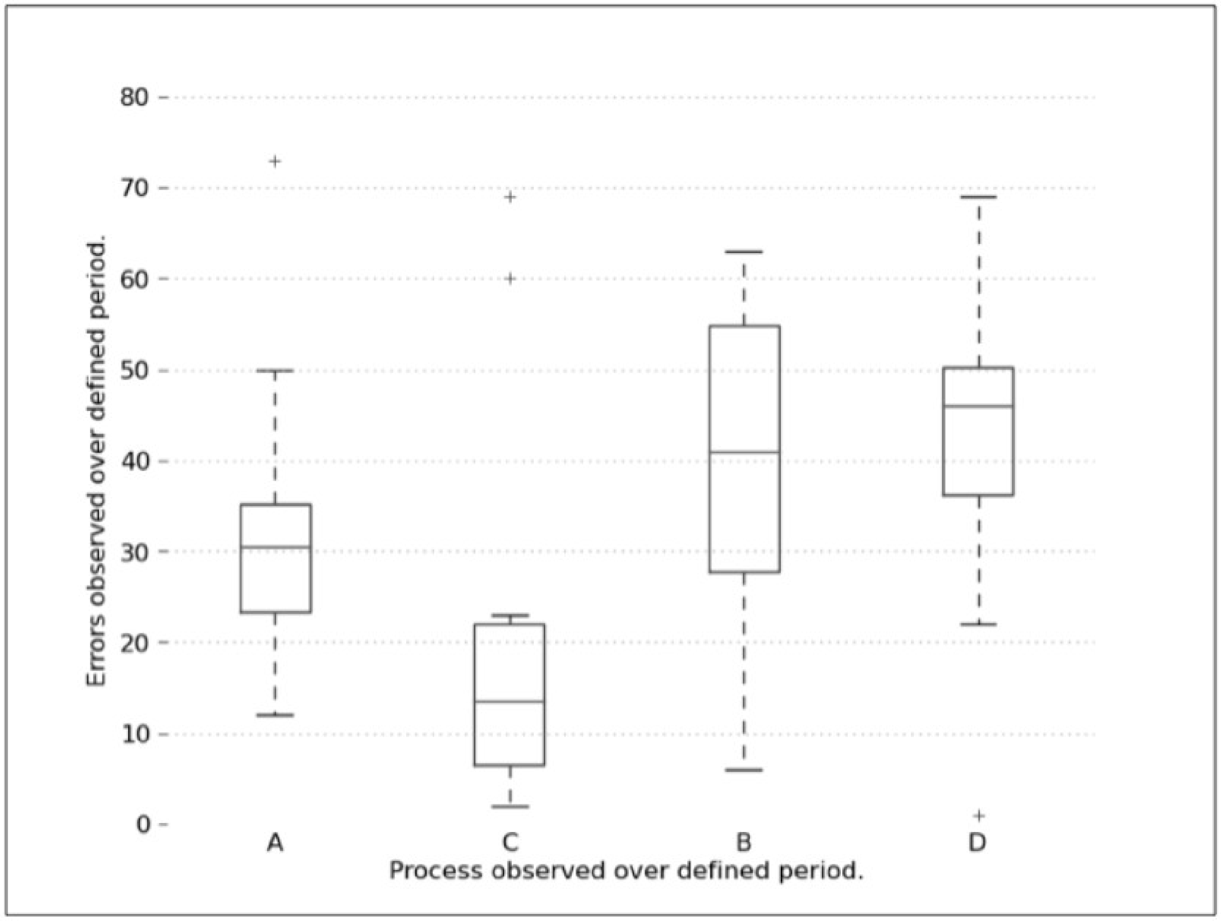
7.[illegible]

[illegible]

```
import matplotlib.pyplot as plt
# define data
PROCESSES = {
   "A": [12, 15, 23, 24, 30, 31, 33, 36, 50, 73],
   "B": [6, 22, 26, 33, 35, 47, 54, 55, 62, 63],
   "C": [2, 3, 6, 8, 13, 14, 19, 23, 60, 69],
   "D": [1, 22, 36, 37, 45, 47, 48, 51, 52, 69],
   }
DATA = PROCESSES.values()
LABELS = PROCESSES.keys()
plt.boxplot(DATA, notch=False, widths=0.3)
# set ticklabel to process name
plt.gca().xaxis.set_ticklabels(LABELS)
# some clean up(removing chartjunk)
# turn the spine off
for spine in plt.gca().spines.values():
   spine.set_visible(False)
# turn all ticks for x-axis off
plt.gca().xaxis.set_ticks_position('none')
```

```
# leave left ticks for y-axis on
plt.gca().yaxis.set_ticks_position('left')
# set axes labels
plt.ylabel("Errors observed over defined period.")
plt.xlabel("Process observed over defined period.")
plt.show()
```

□□□□□□□□□□□□8-4□□□



□8-4

8.3.3 □□□□

□□□□□□□□□□DATA□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□

[illegible]“[illegible]”[illegible] Edward R. Tufte [illegible]The Visual Display of Quantitative Information [illegible]

8.4 [illegible]

[illegible] Henry Gantt[illegible]19 [illegible]10[illegible]

[illegible]

[illegible]x[illegible]y[illegible]

[illegible]

[illegible]

[illegible]Python[illegible]

8.4.1 [illegible]

[illegible] Python[illegible]

[illegible]

8.4.2 □□□□

□□□□□□□□□□□□□□□□□□Python□matplotlib□□□□□□□□□□□□□□

1.□□□□□□□TEST_DATA□□□TEST_DATA□□□Gantt□□

2.□□□□□□□□□□□□□□□□□□□□□

3.□□□□□□□□□□□□□□□□□□□

4.□□□□□□□□□x□□□y□□

5.□□□□□□□□□

6.□□□□□□

□□□□□□□□

```
from datetime import datetime
import sys
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.font_manager as font_manager
import matplotlib.dates as mdates
import logging
class Gantt(object):
    '''
    Simple Gantt renderer.
    Uses *matplotlib* rendering capabilities.
    '''
    # Red Yellow Green diverging colormap
    # from http://colorbrewer2.org/
    RdYlGr = ['#d73027', '#f46d43', '#fdae61',
        '#fee08b', '#ffffbf', '#d9ef8b',
        '#a6d96a', '#66bd63', '#1a9850']
    POS_START = 1.0
    POS_STEP = 0.5
    def __init__(self, tasks):
        self._fig = plt.figure()
        self._ax = self._fig.add_axes([0.1, 0.1, .75, .5])
        self.tasks = tasks[::-1]
    def _format_date(self, date_string):
        '''
        Formats string representation of *date_string* into
        *matplotlib. dates*
        instance.
        '''
        try:
            date = datetime.strptime(date_string, '%Y-%m-%d
    %H:%M:%S')
        except ValueError as err:
            logging.error("String '{0}' can not be converted to
            datetime object: {1}"
                .format(date_string, err))
            sys.exit(-1)
        mpl_date = mdates.date2num(date)
        return mpl_date
    def _plot_bars(self):
        '''
        Processes each task and adds *barh* to the current
  *self._ax*(*axes*).
        '''
```

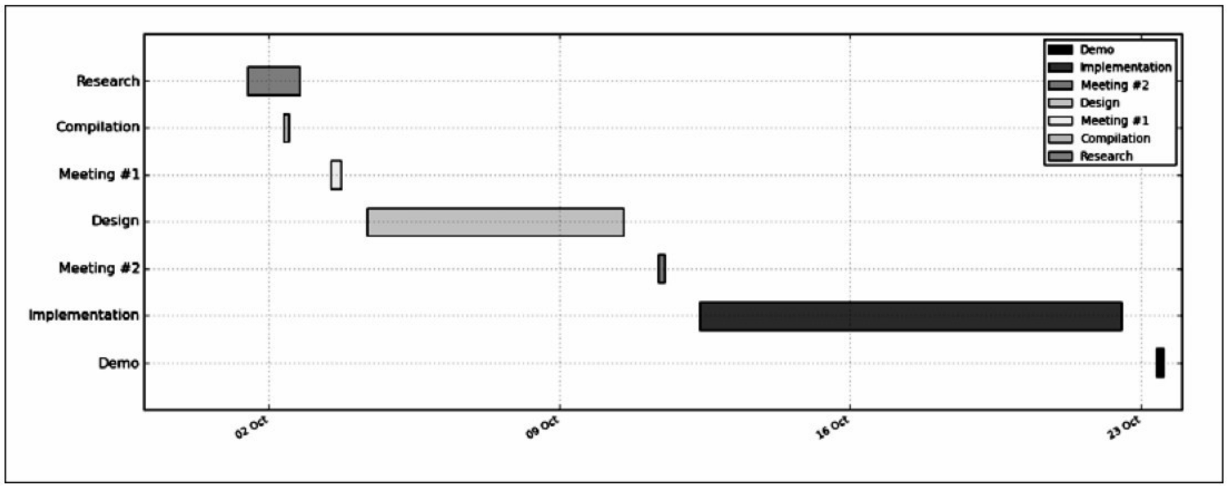
```
      i = 0
      for task in self.tasks:
         start = self._format_date(task['start'])
         end = self._format_date(task['end'])
         bottom = (i * Gantt.POS_STEP) + Gantt.POS_START
         width = end - start
         self._ax.barh(bottom, width, left=start, height=0.3,
            align='center', label=task['label'],
            color = Gantt.RdYlGr[i])
         i += 1
   def _configure_yaxis(self):
      '''y axis'''
      task_labels = [t['label'] for t in self.tasks]
      pos = self._positions(len(task_labels))
      ylocs = self._ax.set_yticks(pos)
      ylabels = self._ax.set_yticklabels(task_labels)
      plt.setp(ylabels, size='medium')
   def _configure_xaxis(self):
      '''x axis'''
      # make x axis date axis
      self._ax.xaxis_date()
      # format date to ticks on every 7 days
      rule        =        mdates.rrulewrapper(mdates.DAILY,
interval=7)
      loc = mdates.RRuleLocator(rule)
      formatter = mdates.DateFormatter("%d %b")
      self._ax.xaxis.set_major_locator(loc)
```

```
        self._ax.xaxis.set_major_formatter(formatter)
        xlabels = self._ax.get_xticklabels()
        plt.setp(xlabels, rotation=30, fontsize=9)
    def _configure_figure(self):
        self._configure_xaxis()
        self._configure_yaxis()
        self._ax.grid(True, color='gray')
        self._set_legend()
        self._fig.autofmt_xdate()
    def _set_legend(self):
        '''
        Tweak font to be small and place *legend*
        in the upper right corner of the figure
        '''
        font = font_manager.FontProperties(size='small')
        self._ax.legend(loc='upper right', prop=font)
    def _positions(self, count):
        '''
        For given *count* number of positions, get array for
the
    positions.
        '''
        end = count * Gantt.POS_STEP + Gantt.POS_START
        pos      =      np.arange(Gantt.POS_START,      end,
Gantt.POS_STEP)
        return pos
```

□□
□□□□□□□□□□□□□x □□□□□□□□□□□□□ y □□□□□□□□□□□□□

```
def show(self):
    self._plot_bars()
    self._configure_figure()
    plt.show()
if __name__ == '__main__':
    TEST_DATA = (
        { 'label': 'Research',       'start':'2013-10-01
        12:00:00',     'end':     '2013-10-02     18:00:00'},     #
@IgnorePep8
        { 'label': 'Compilation',    'start':'2013-10-02
        09:00:00',     'end':     '2013-10-02     12:00:00'},     #
@IgnorePep8
        { 'label': 'Meeting #1',    'start':'2013-10-03
        12:00:00',     'end':     '2013-10-03     18:00:00'},     #
@IgnorePep8
        { 'label': 'Design',       'start':'2013-10-04
        09:00:00',     'end':     '2013-10-10     13:00:00'},     #
@IgnorePep8
        { 'label': 'Meeting #2',    'start':'2013-10-11
        09:00:00',     'end':     '2013-10-11     13:00:00'},     #
@IgnorePep8
        { 'label': 'Implementation', 'start':'2013-10-12
        09:00:00',     'end':     '2013-10-22     13:00:00'},     #
@IgnorePep8
        { 'label': 'Demo',       'start':'2013-10-23
    09:00:00',    'end':    '2013-10-23    13:00:00'},    #
@IgnorePep8
    )
  gantt = Gantt(TEST_DATA)
  gantt.show()
```

□□□□□□□□□□□□□□□□□□8-5□□□



□8-5

8.4.3 □□□□

□□□□□□□□□□□"__main__"□ if □□□□□□□□□□□□□ TEST_DATA□□□□□ Gantt □□□□□□□□□□□□□□□□□□□□□□□□ TASK_DATA[3] □□□self.tasks□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□show()□□□□□□□□□□□□□□□□□□□□

```
def show(self):
  self._plot_bars()
  self._configure_figure()
  plt.show()
```

[illegible] matplotlib.pyplot.barh [illegible] self._ax [illegible]bottom[illegible]

[illegible]colorbrewer2. org[illegible]divergent[illegible]

[illegible] x [illegible] y [illegible] matplotlib.pyplot.barh[illegible]

[illegible]grid[illegible]legend[illegible]

[illegible]plt.show()[illegible]

8.5 [illegible]

[illegible]

8.5.1 [illegible]

[illegible]

[illegible]95%[illegible]

matplotlib[illegible]matplotlib.pyplot.errorbar[illegible]

[illegible]yerr[illegible]xerr[illegible]

8.5.2 [illegible]

[illegible]

1.[illegible]

2.[illegible]

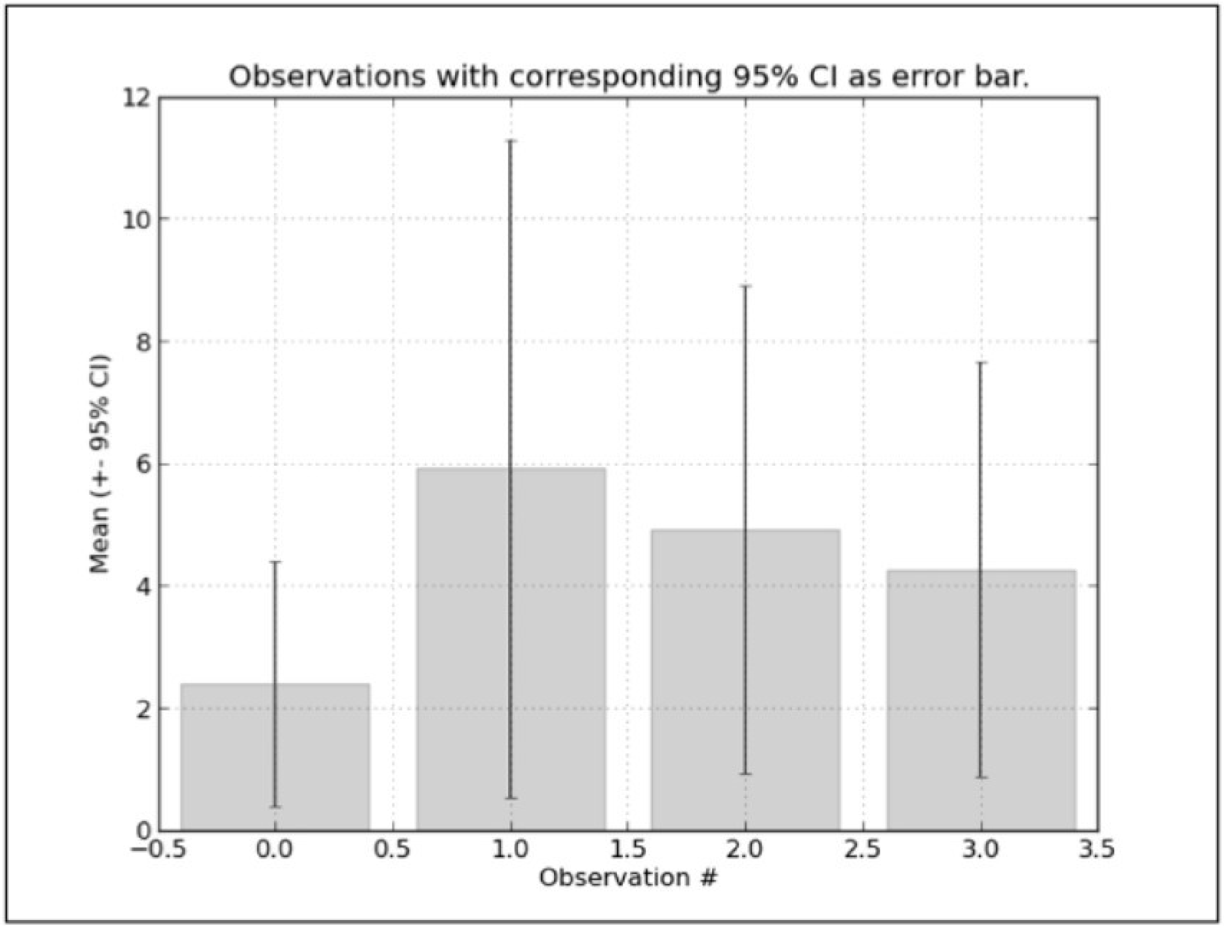
3.[illegible]95%[illegible]

4.[illegible]

[illegible]

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as sc
TEST_DATA = np.array([[1,2,3,2,1,2,3,4,2,3,2,1,2,3,4,4,3,2,3,2,3,2,1],
    [5,6,5,4,5,6,7,7,6,7,7,2,8,7,6,5,5,6,7,7,7,6,5],
    [9,8,7,8,8,7,4,6,6,5,4,3,2,2,2,3,3,4,5,5,5,6,1],
    [3,2,3,2,2,2,2,3,3,3,3,4,4,4,4,5,6,6,7,8,9,8,5],
    ])
# find mean for each of our observations
y = np.mean(TEST_DATA, axis=1, dtype=np.float64)
# and the 95% confidence interval
ci95 = np.abs(y - 1.96 * sc.sem(TEST_DATA, axis=1))
# each set is one try
tries = np.arange(0, len(y), 1.0)
# tweak grid and setup labels, limits
plt.grid(True, alpha=0.5)
plt.gca().set_xlabel('Observation #')
plt.gca().set_ylabel('Mean (+- 95% CI)')
plt.title("Observations with corresponding 95% CI as error bar.")
plt.bar(tries, y, align='center', alpha=0.2)
plt.errorbar(tries, y, yerr=ci95, fmt=None)
plt.show()
```

□□□□□□□□□□□□□□□□95%□□□□□y□□□□□□□□□□□□□□□□□□□□□□□□□□□8-6□□□□□□□□



□8-6

8.5.3 □□□□

□□□□□□□□□□□□□□□□□□NumPy□□□□□□□□□□□□□□□□□□□□□□□□□□

NumPy□□□□□□C□□□□□Python□□□□□□□□□□□□□□□

[illegible]NumPy[illegible]

[illegible]np.mean[illegible]dtype=np.float64[illegible] NumPy [illegible] http://docs.scipy.org/doc/numpy/reference/generated/ numpy.mean.html[illegible]np.mean[illegible] np.float32[illegible]np.float64[illegible]

8.5.4 [illegible]

[illegible] SD[illegible]2SD[illegible]SE [illegible]95%CI[illegible]

[illegible]Standard Deviation[illegible]68.2%[illegible]2/3[illegible]±SD[illegible]95.4%[illegible]±2*SD [illegible]

[illegible]Standard Error[illegible] SD [illegible] N [illegible]SD/√N[illegible] N [illegible]SE[illegible]

[illegible]SE[illegible]95%[illegible]/[illegible] 1.96*SE[illegible] 95% CI = M ± (1.96 * SE)[illegible]

[illegible]

8.6 [illegible]

□□matplotlib□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□LaTex□□□□□“□LaTex□□□□”□□□□□□□□□

8.6.1 □□□□

□□□□□ matplotlib □□□□□□□□□□□□□□□□□□□□□□pyplot □□

□ 8-1 □□□□□□□□□□□□□□ matplotlib OO API □□□□□□

□8-1

matplotlib.pyplot	Matplotlib API	描　　述
text	matplotlib.axes.Axes.text	在指定的位置（x，y）为坐标轴添加文本。fontdict 参数允许我们覆盖一般的字体属性，或者可以使用 kwargs 覆盖特定的属性
xlabel	matplotlib.axes.Axes.set_xlabel	设置 x 轴的标签。通过 labelpad 指定标签和 x 坐标轴之间的间隔

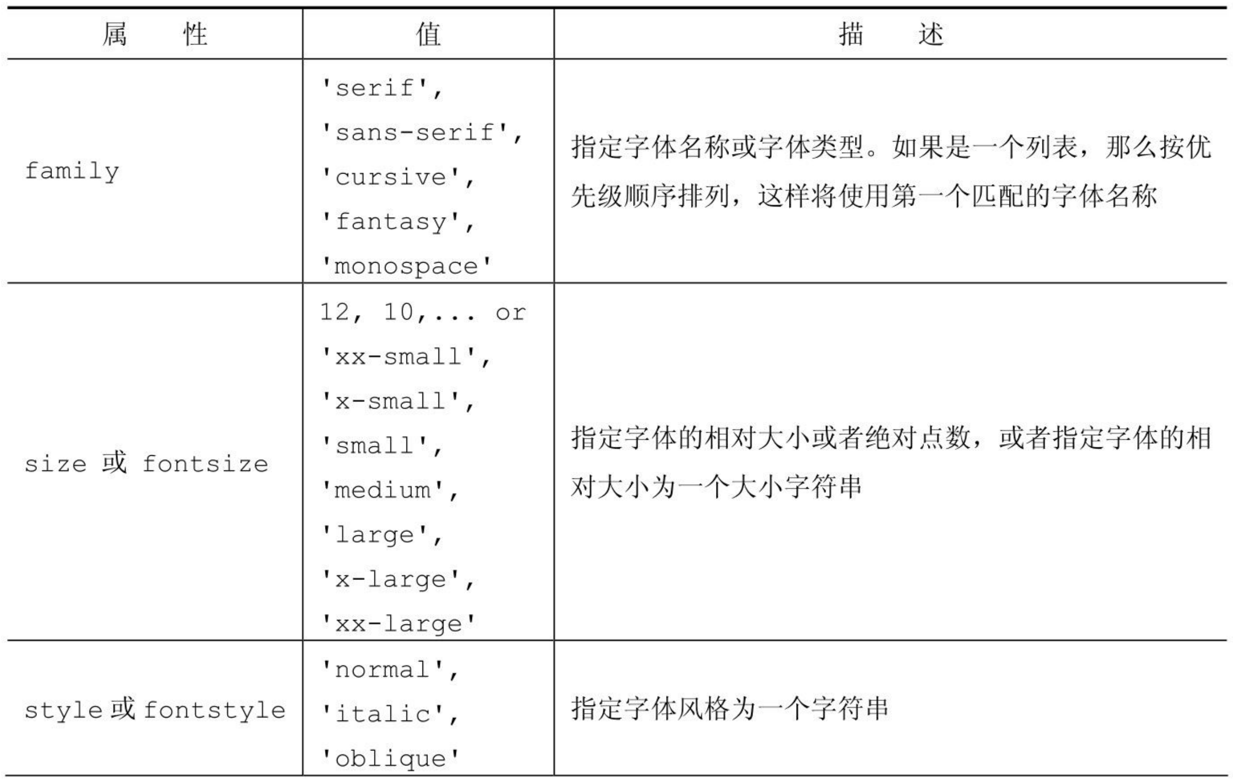
□□

matplotlib.pyplot	Matplotlib API	描　　述
ylabel	matplotlib.axes.Axes.set_ylabel	和 xlabel 类似，但用于 y 轴
title	matplotlib.axes.Axes.set_title	设置坐标轴的标题。接受所有一般的文本属性，如 fontdict 和 kwargs
suptitle	matplotlib.figure.Figure.suptitle	为图表添加一个居中的标题。通过 kwargs 接受所有通用文本属性。使用 Figure 坐标
figtext	matplotlib.figure.Figure.text	在图表的任意位置添加文本。位置通过 x、y 定义，使用图表的归一化坐标。使用 fontdict 覆盖字体属性，但也支持使用 kwargs 覆盖任何文本相关的属性

□□□□□□□□□□□□□□□□□□□□matplotlib.text.Text□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

matplotlib.text.Text□□□□□□□□□8-2□□□

□8-2



属　　性	值	描　　述
family	'serif', 'sans-serif', 'cursive', 'fantasy', 'monospace'	指定字体名称或字体类型。如果是一个列表，那么按优先级顺序排列，这样将使用第一个匹配的字体名称
size 或 fontsize	12, 10,... or 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'	指定字体的相对大小或者绝对点数，或者指定字体的相对大小为一个大小字符串
style 或 fontstyle	'normal', 'italic', 'oblique'	指定字体风格为一个字符串

□□

属　　性	值	描　　述
variant	'normal', 'small-caps'	指定字体的变体形式
weight或fontweight	0-1000 or 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extrabold', 'black'	指定字体粗细或者使用一个特定的粗细字符串。字体粗细定义为相对于字体高度的字符轮廓厚度
stretch或fontstretch	0-1000 or 'ultra-condensed', 'extra-condensed', 'condensed', 'semi-condensed', 'normal', 'semi-expanded', 'expanded', 'extra-expanded', 'ultra-expanded'	指定字体的拉伸。拉伸定义为水平的压缩或者扩张。该属性目前没有实现
fontproperties		默认使用 matplotlib.font_manager. FontProperties 实例。该类存储并管理 W3C CSS Level1 规范中描述的字体属性。规范网址为 http://www.w3.org/ TR/1998/REC-CSS2- 19980512/

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

[illegible]rcParams['text.color'][illegible]

[illegible]

◆ horizontalalignment [illegible] ha[illegible] center[illegible]left[illegible]right[illegible]

◆ verticalalignment [illegible] va[illegible] center[illegible]top[illegible]bottom[illegible]baseline[illegible]

◆ multialignment[illegible] left[illegible]right[illegible]center[illegible]

8.6.2 [illegible]

[illegible]

1.[illegible]

2.[illegible]

3.[illegible]

4.[illegible]

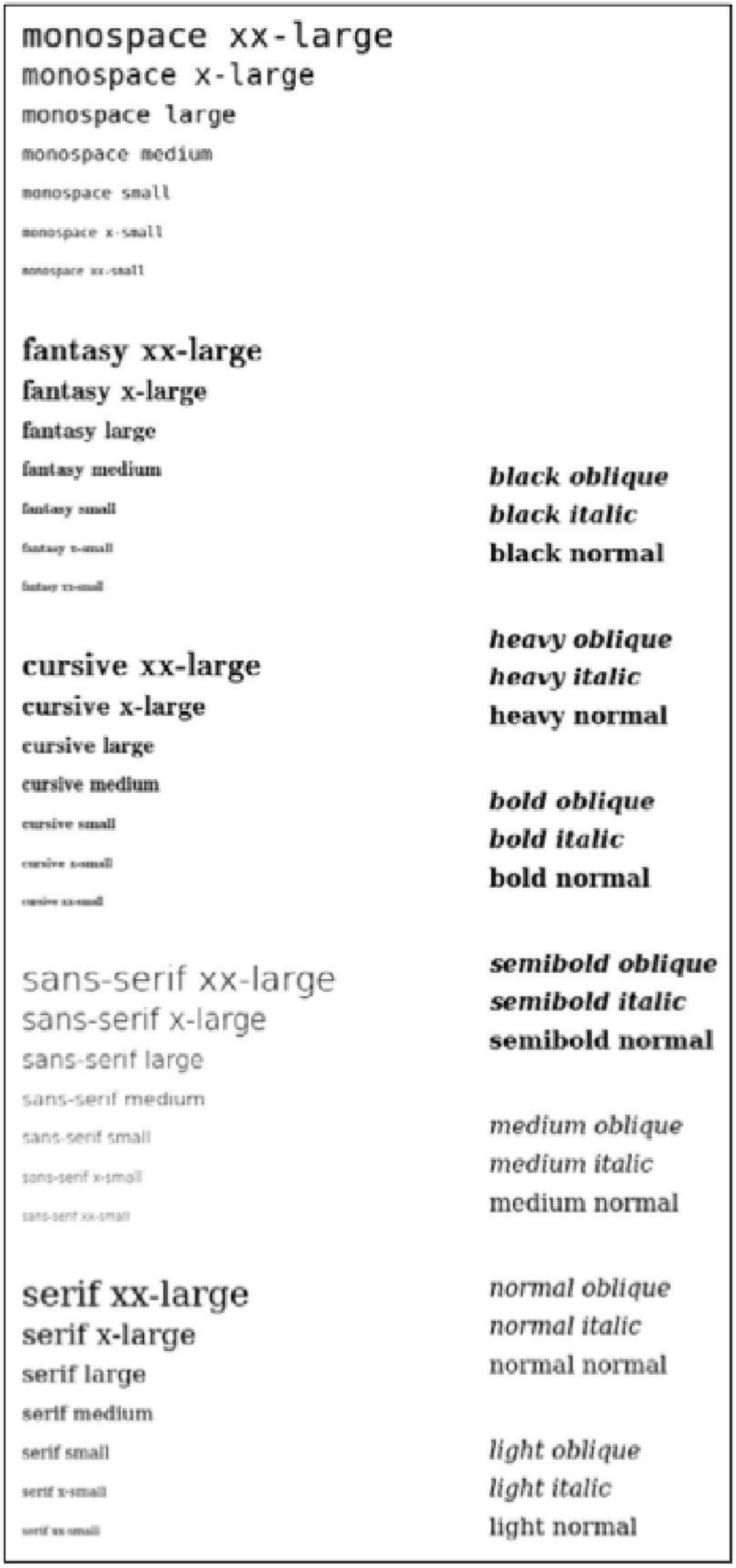
5.[illegible]

[illegible]

```
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
# properties:
families = ['serif', 'sans-serif', 'cursive', 'fantasy', 'monospace']
sizes = ['xx-small', 'x-small', 'small', 'medium', 'large',
    'x-large', 'xx-large']
    styles = ['normal', 'italic', 'oblique']
    weights = ['light', 'normal', 'medium', 'semibold', 'bold',
'heavy',
    'black']
    variants = ['normal', 'small-caps']
    fig = plt.figure(figsize=(9,17))
    ax = fig.add_subplot(111)
    ax.set_xlim(0,9)
    ax.set_ylim(0,17)
      # VAR: FAMILY, SIZE
    y = 0
    size = sizes[0]
    style = styles[0]
    weight = weights[0]
    variant = variants[0]
    for family in families:
      x = 0
      y = y + .5
      for size in sizes:
        y = y + .4
        sample = family + " " + size
        ax.text(x, y, sample, family=family, size=size,
          style=style, weight=weight, variant=variant)
    # VAR: STYLE, WEIGHT
    y = 0
    family = families[0]
    size = sizes[4]
```

```
variant = variants[0]
for weight in weights:
    x = 5
    y = y + .5
    for style in styles:
        y = y + .4
        sample = weight + " " + style
        ax.text(x, y, sample, family=family, size=size,
            style=style, weight=weight, variant=variant)
ax.set_axis_off()
plt.show()
```

□□□□□□□□□8-7□□□□□□



monospace xx-large
monospace x-large
monospace large
monospace medium
monospace small
monospace x-small
monospace xx-small

fantasy xx-large
fantasy x-large
fantasy large
fantasy medium
fantasy small
fantasy x-small
fantasy xx-small

cursive xx-large
cursive x-large
cursive large
cursive medium
cursive small
cursive x-small
cursive xx-small

sans-serif xx-large
sans-serif x-large
sans-serif large
sans-serif medium
sans-serif small
sans-serif x-small
sans-serif xx-small

serif xx-large
serif x-large
serif large
serif medium
serif small
serif x-small
serif xx-small

black oblique
black italic
black normal

heavy oblique
heavy italic
heavy normal

bold oblique
bold italic
bold normal

semibold oblique
semibold italic
semibold normal

medium oblique
medium italic
medium normal

normal oblique
normal italic
normal normal

light oblique
light italic
light normal

[illegible]8-7

8.6.3 [illegible]

[illegible]

[illegible]

[illegible]matplotlib[illegible] Ubuntu 13.04 [illegible]

8.7 [illegible]LaTeX[illegible]

[illegible]

[illegible] matplotlib [illegible] LaTex [illegible]

LaTeX [illegible]

LaTeX [illegible]

```
\documentclass{article}
\title{This here is a title of my document}
\author{Peter J. S. Smith}
\date{September 2013}
\begin{document}
   \maketitle
```

Hello world, from LaTeX!

\end{document}

[illegible] WYSIWYG [4] [illegible]

[illegible] LaTeX [illegible]

[illegible] LaTeX [illegible] http://latex-project.org/ [illegible]

8.7.1 [illegible]

[illegible] matplotlib [illegible] LaTeX [illegible]

◆ LaTeX system [illegible] TeX Live [illegible]

◆ DVI to PNG converter [illegible] TeX [illegible] DVI [illegible] PNG [illegible]

◆ Ghost script [illegible] TeX Live [illegible]

[illegible] LaTeX [illegible] Linux [illegible] TeX Live [illegible] TeX [illegible] Mac OS [illegible] MacTeX [illegible] Windows [illegible] proTeX [illegible] TeX [illegible] LaTeX [illegible]

[illegible] TeX [illegible]

[illegible] Linxu [illegible] Ubuntu [illegible] textlive [illegible] dvipng [illegible]

$ sudo apt-get install texlive dvipng

[illegible] text.usetex [illegible] True [illegible] matplotlib [illegible] LaTeX [illegible] .matplotlibrc [illegible] rcParams['text'] [illegible] Unix [illegible] /home/<user>/.matplotlibrc [illegible] Windows [illegible]

C:\Documents and Settings\<user>\. matplotlibrc□□□□□□□□□□
□□□□□□

```
matplotlib.pyplot.rc('text', usetex=True)
```

□□□□□□□□□□matplotlib□□□□□□□□□□□□LaTeX□□□□□□□□□□□□
□□□□□□□□□□□□□□

□□□□□□□□□□□LaTeX□□□□□Agg□PS□PDF□□□□□□LaTeX□□□□□

8.7.2 □□□□

□□□□□□□LaTeX□□□□□□□□□□□□□

1.□□□□□□□□□

2.□□□□□□session□□□matplotlib□□LaTeX□

3.□□□□□□□□□□□□□□

4.□□□□□□□

5.□□□□□□□□□□□□

6.□□□□□□□□□□□□□

7.□□□□□□□□□□□□□

8.□□□□□□□□□□□

9.□□□□□□□□□□□□□□□□

10.□x□□y□□□□□□□□□□□□□□□□□□□□□

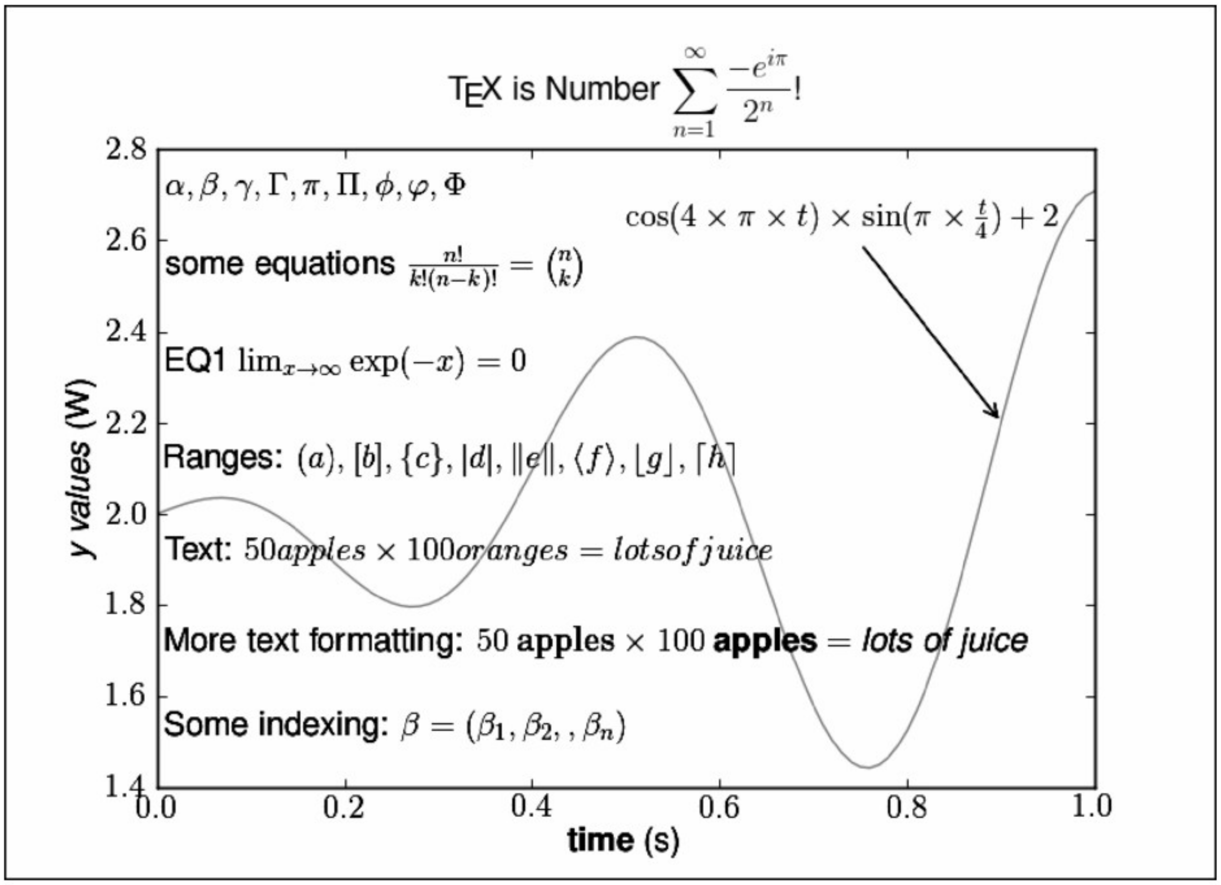
□□□□□□□□□□□□□

```
import numpy as np
import matplotlib.pyplot as plt
# Example data
t = np.arange(0.0, 1.0 + 0.01, 0.01)
s = np.cos(4 * np.pi * t) * np.sin(np.pi*t/4) + 2
plt.rc('text', usetex=True)
```

```
    plt.rc('font',**{'family':'sans-serif','sans-serif':
['Helvetica'],
    'size':16})
    plt.plot(t, s, alpha=0.25)
    # first, the equation for 's'
    # note the usage of Python's raw strings
    plt.annotate(r'$\cos(4 \times \pi \times {t}) \times \sin(\pi
\times \frac{t} 4) + 2$', xy=(.9,2.2), xytext=(.5, 2.6),
color='red', arrowprops=
    {'arrowstyle':'->'})
    # some math alphabet
    plt.text(.01, 2.7, r'$\alpha, \beta, \gamma, \Gamma, \pi,
\Pi, \phi, \varphi, \Phi$')
    # some equation
    plt.text(.01, 2.5, r'some equations $\frac{n!}{k!(n-k)!}=
{n \choose k}$')
    # more equations
    plt.text(.01, 2.3, r'EQ1 $\lim_{x \to \infty} \exp(-x) = 0$')
    # some ranges...
    plt.text(.01, 2.1, r'Ranges: $( a ), [ b ], \{ c \}, | d |, \| e \|,
\langle f \rangle, \lfloor g \rfloor, \lceil h \rceil$')
    # you can multiply apples and oranges
    plt.text(.01, 1.9, r'Text:$50 apples \times 100 oranges =
lots of juice$')
    plt.text(.01, 1.7, r'More text formatting:$50 \textrm{
apples}\times 100\textbf{ apples} = \textit{lots of juice}$')
```

```
    plt.text(.01, 1.5, r'Some indexing: $\beta =
(\beta_1,\beta_2,\dotsc, \beta_n)$')
    # we can also write on labels
    plt.xlabel(r'\textbf{time} (s)')
    plt.ylabel(r'\textit{y values} (W)')
    # and write titles using LaTeX
    plt.title(r"\TeX\ is Number "
      r"$\displaystyle\sum_{n=1}^\infty\frac{-e^{i\pi}}
  {2^n}$!",
      fontsize=16, color='gray')
    # Make room for the ridiculously large title.
    plt.subplots_adjust(top=0.8)
    plt.savefig('tex_demo')
    plt.show()
```

□□□□□□□□□□8-8□□□□□□□□□□□□□□LaTeX□□□□□□



□8-8

8.7.3 □□□□

□□□□□□□□□□□□□□□□□□□□ matplotlib□□□□□□□□□ matplotlib.pyplot.annotate □ matplotlib.pyplot.text □ matplotlib.pyplot.xlabel □ matplotlib.pyplot.ylabel □ matplotlib.pyplot. title□

□□□□□□□□□□□□□□□□□□Python□□□□□□□□□□□□□□□□LaTeX□□□□□□□□□□□□□□

TeX□□□□□matplotlib□□□□□□□□□□matplotlib□□□□□□ □ □ □ http://matplotlib.org/users/mathtext.html#writing-mathematical-expressions□

□□□□□URL□□LaTeX□□□□□□matplotlib□□□□□TeX□□□□□□□□□□□□□□□□LaTeX□□□□□□□□□□□□□□□

8.7.4 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□LaTeX□□□□□□□□□□□□□□□□□□□□$PATH□□□□□□□Windows□□□□□□□□□□□□□□□□□□□□□□□□□□matplotlib□□□LaTeX□□□□□□

□ matplotlib □ □ □ □ http://matplotlib.org/users/usetex.html#possible-hangups □ LaTeX □□□□ http://tex.stackex change. com/□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

8.8 □□pyplot□OO API□□□

□□□□□□□□□□□□□matplotlib□□□□□□□pyplot□□□□□□API □□□

8.8.1 □□□□

□□□matplotlib□□□□□□□□□——□□□□□□□□□□□□□□□□□□□□□□ □ □ □ □ □ □ □ □ □ MATLAB® □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □http://www.aosabook.org/en/matplotlib.html□□□□□□□□□□□□MATLAB®□ Python□□□□□□□□□□□□□matplotlib□□□□□□□□□□□□□□

□□□ matplotlib □□□□ MATLAB®□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□matplotlib□□□□□□□□□□□□□□□□□□□□□□Linux

[illegible]GTK[illegible]QT[illegible]Tk[illegible]Linux[illegible]Windows[illegible]wxWidgets[illegible]
Cocoa [illegible] Mac OS [illegible] matplotlib [illegible]
matplotlib.pyplot [illegible]plot[illegible]

matplotlib.pyplot[illegible]matplotlib[illegible] plt.plot([1,2,3,4, 5]);plt.show()[illegible]

[illegible]FontProperties[illegible]AxesGrid[illegible]matplotlib.pyplot[illegible]

[illegible] OO API[illegible]

[illegible]matplotlib[illegible]

◆ matplotlib.pylab [illegible] MATLAB®[illegible]

◆ matplotlib API[illegible] matplotlib [illegible]

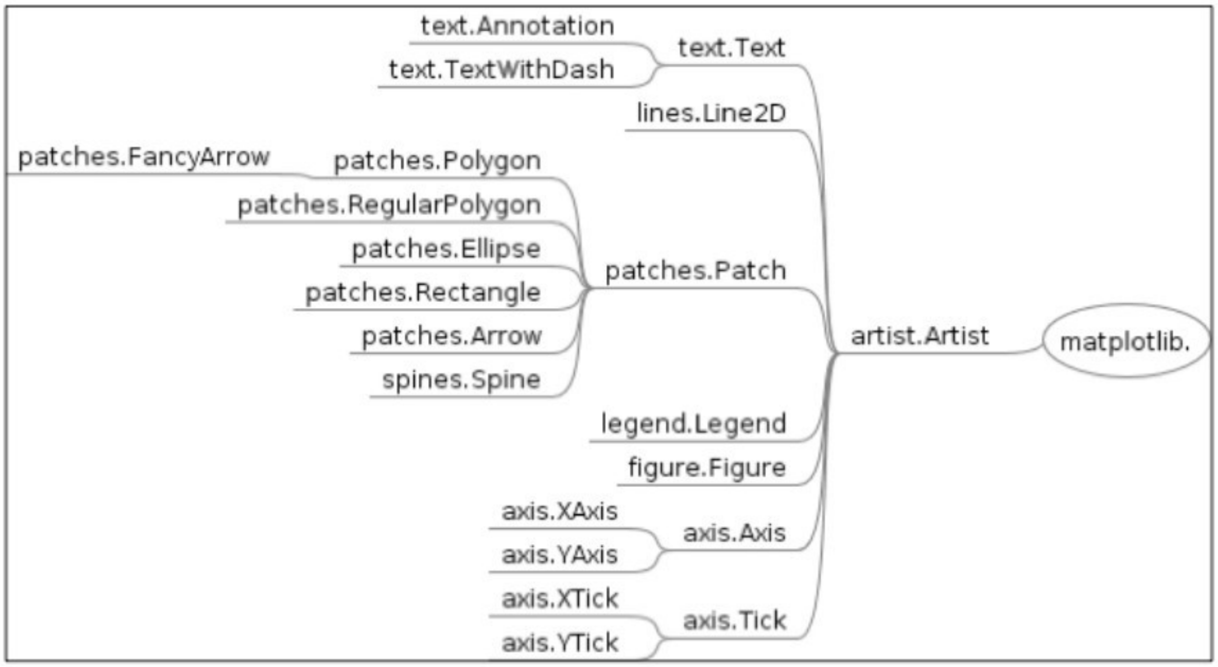
◆ [illegible]

[illegible]FigureCanvas[illegible]Renderer[illegible]Event[illegible]

[illegible] matplotlib.backend_bases [illegible]GTK 3[illegible]matplotlib.backends.backend_gkt3agg [5] [illegible]

[illegible] Artist [illegible]Artist[illegible]Renderer[illegible]FigureCanvas[illegible]Artist[illegible]Artist[illegible]matplotlib.artist[illegible]

matplotlib.artist.Artist □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□8-9□□□



□8-9

□□□□□□□Artist □□□□□□□□□□□□□□□□□□□□□□□Artist□□□□□□□□□□ artists□□□□□□□□□□ Line2D□Rectangle□Circle□Text□□□□□□□□artists□□□□Artists□□□□Axis□Tick□Axes□Figure□□□□Figure□□□□artist——Rectangle□□□□□□□□□□□□□□□artist——Axes□

□□□□□□□□□□Axes□□matplotlib.axes.Axes□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Axes□□Axes□□□□□□□□□□□□□□helper□□□□□□□□□□□artist□□□□□□□□Axes□□□□□plot□hist□imshow□

□□□□□Axes.hist □□□□□ matplotlib.patch.Rectangle □□□□□□□□□□Axes.patches□□□□

Axes.plot□□□□□□□□□matplotlib.lines.Line2D□□□□□□□□□□□□Axes.lines□□□□

8.8.2 □□□□

□□□□□□□□□□□□

1.□□□□□□□□□□□□□ matplotlib Path □□□

2.□□□□□□□□

3.□□□□□□□□□□□□□□□□□□□

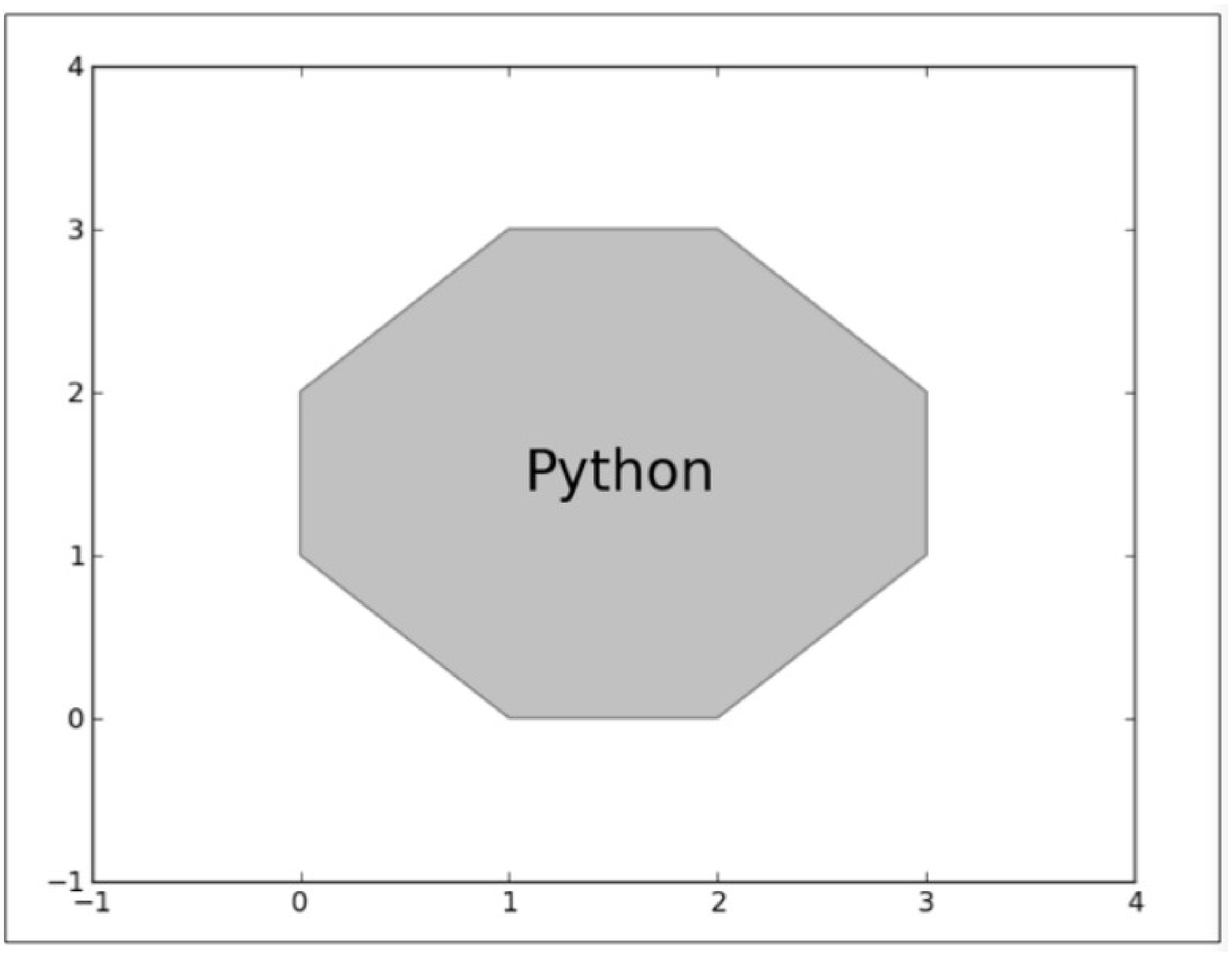
4.□□□□□□□

5.□□□□□figure□Axes□□□□

□□□□□□□□

```
import matplotlib.pyplot as plt
from matplotlib.path import Path
import matplotlib.patches as patches
# add figure and axes
fig = plt.figure()
ax = fig.add_subplot(111)
coords = [
    (1., 0.), # start position
    (0., 1.),
    (0., 2.), # left side
    (1., 3.),
    (2., 3.),
    (3., 2.), # top right corner
    (3., 1.), # right side
    (2., 0.),
    (0., 0.), # ignored
    ]
line_cmds = [Path.MOVETO,
    Path.LINETO,
    Path.LINETO,
   Path.LINETO,
   Path.LINETO,
   Path.LINETO,
   Path.LINETO,
   Path.LINETO,
   Path.CLOSEPOLY,
   ]
# construct path
path = Path(coords, line_cmds)
# construct path patch
patch = patches.PathPatch(path, lw=1,
   facecolor='#A1D99B', edgecolor='#31A354')
# add it to *ax* axes
ax.add_patch(patch)
ax.text(1.1, 1.4, 'Python', fontsize=24)
ax.set_xlim(-1, 4)
ax.set_ylim(-1, 4)
plt.show()
```

□□□□□□□□□□□8-10□□□



[illegible]8-10

8.8.3 [illegible]

[illegible]matplotlib.path.Path[illegible]moveto[illegible]lineto[illegible]

[illegible] matplotlib. path.Path [illegible] path [illegible] matplotlib.patched. PathPatch [illegible]——[illegible]

[illegible]fig.axes[illegible]

[illegible]matplotlib.figure.Figure[illegible]matplotlib.pyplot.figure()[illegible]pyplot.figure()[illegible]matplotlibrc[illegible]rc[illegible]figsize[illegible]dpi[illegible]Gcf[illegible]pyplot.figure()[illegible]

[illegible]pyplot[illegible]Figure[illegible]Axes[illegible]Axis[illegible]matplotlib[illegible]

8.8.4 [illegible]

[illegible]Python[illegible]shell[illegible]matplotlib[illegible]IPython pylab [illegible]shell[illegible]matplotlib[illegible]shell [illegible] IPython Notebook[illegible]

IPython Notebook [illegible] Web [illegible] IPython shell [illegible]HTML[illegible]PDF[illegible]Matplotlib[illegible]

[illegible]

[1]. [illegible] 2 [illegible]“[illegible]”[illegible]

[2]. chartjunk [illegible]

[3]. [illegible] TEST_DATA[illegible]

[4]. [illegible]What You See Is What You Get [illegible]

[5]. [illegible] matplotlib.backends.backend_gtk3agg[illegible]